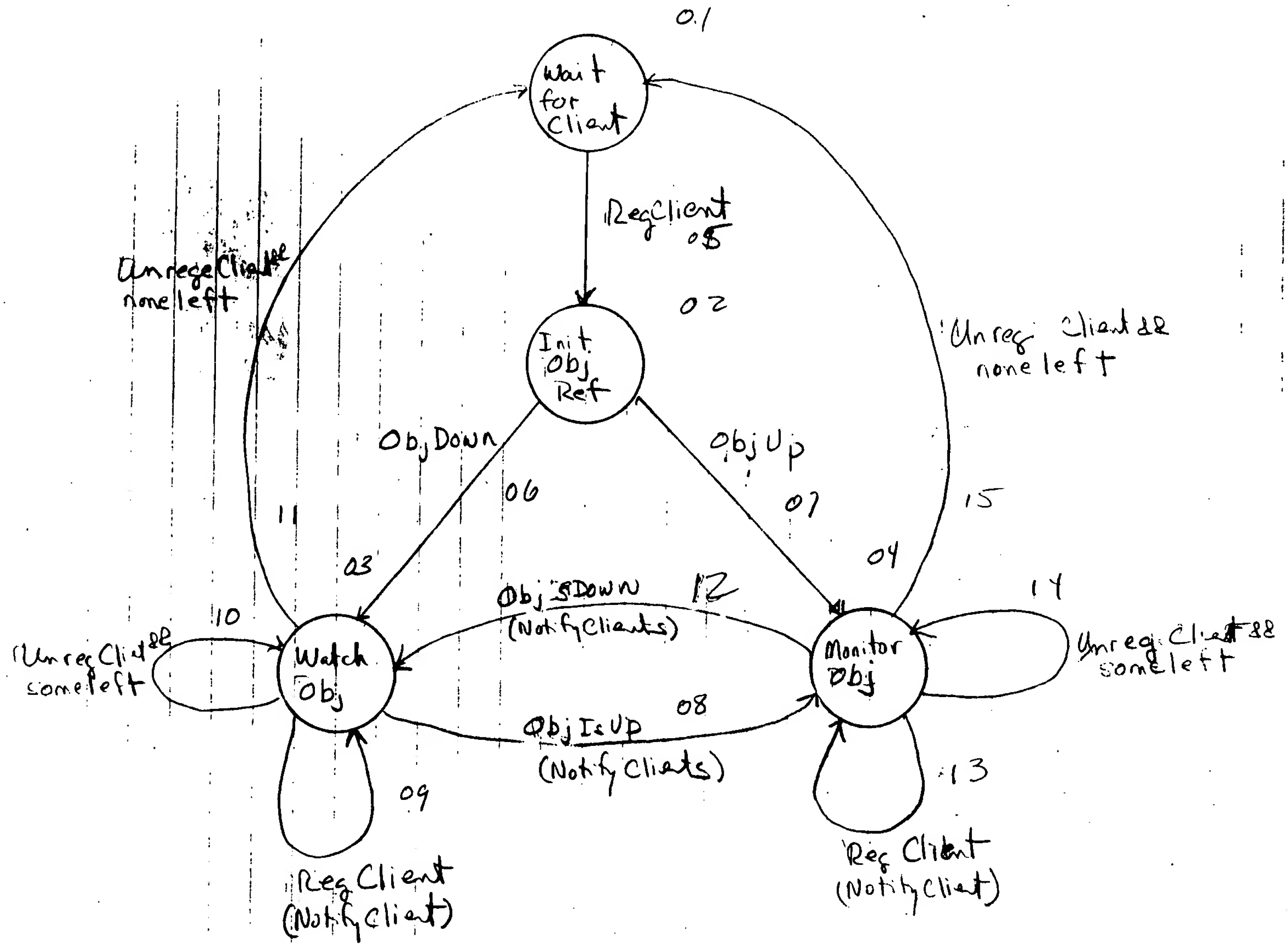

01
Wait for Client
RegClient
05
02
Init Obj Ref
UnregeClient && none left
Unreg Client && none left
Obj Down
06
Obj Up
07
11
15
03
04
Obj IsDown
12
(NotifyClients)
10
UnregClient && some left
Watch Obj
Monitor Obj
14
Unreg Client && some left
Obj IsUp
08
(NotifyClients)
09
13
Reg Client
(NotifyClient)
Reg Client
(NotifyClient)

Fig 1

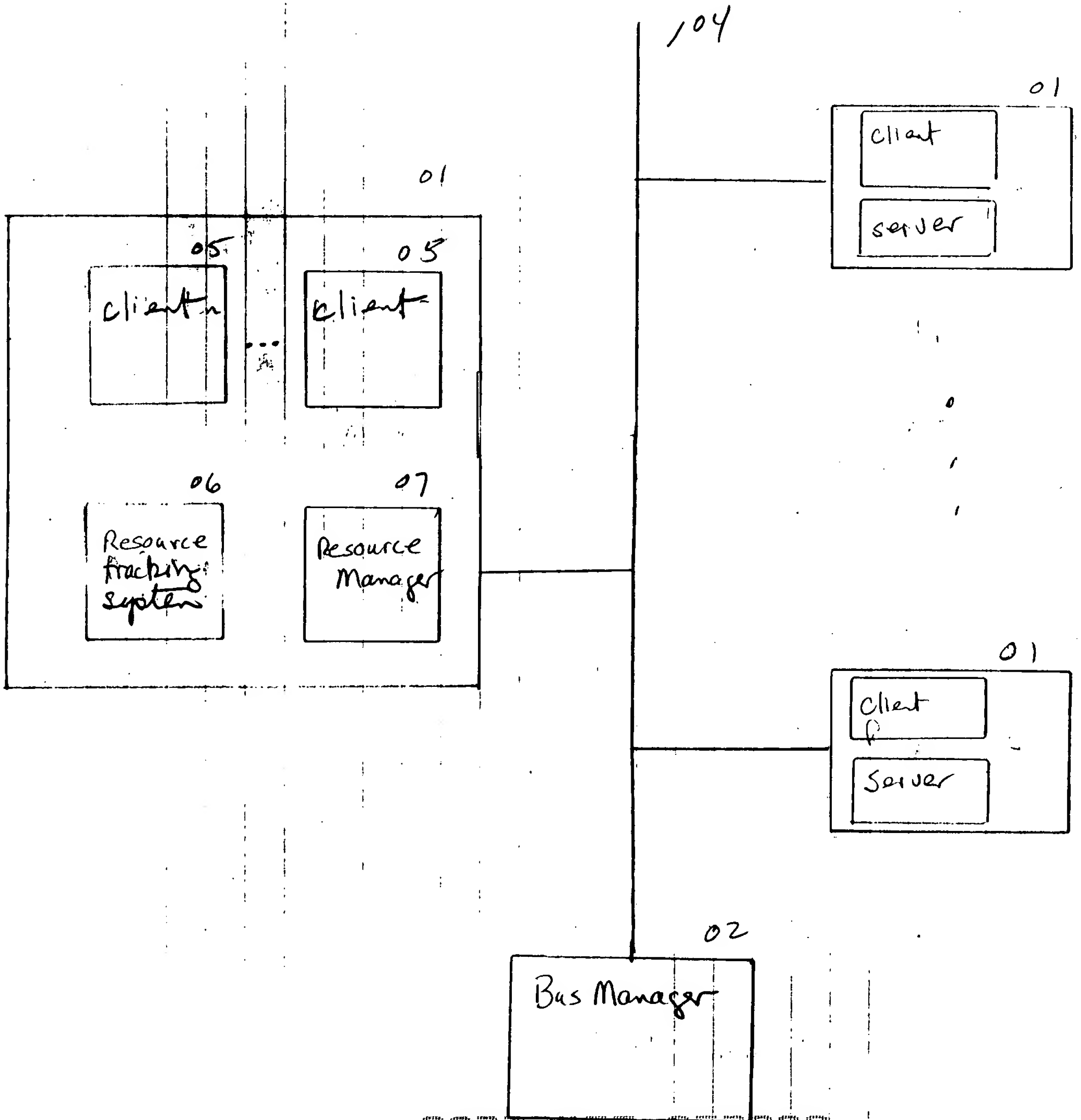
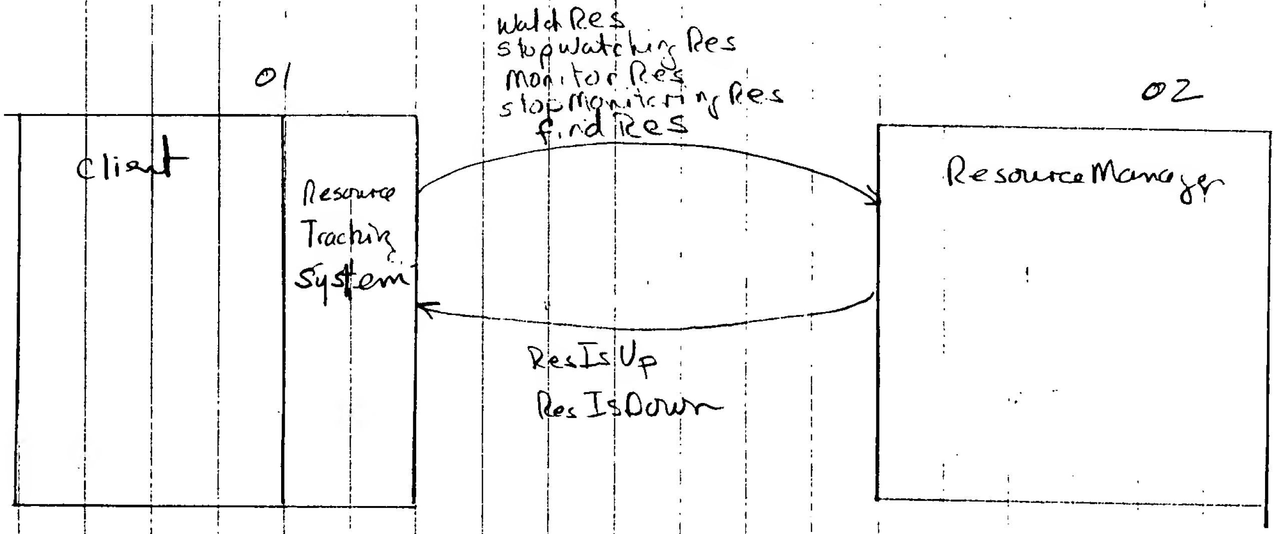

104
01
client n
05
client
05
Resource tracking system
06
Resource Manager
07
01
Cliant
server
01
Cliet
Server
02
Bus Manager

Fig 2


01
Client
Resource Tracking System
WatchRes
stopWatchingRes
MonitorRes
stopMonitoringRes
findRes
ResIsUp
ResIsDown
02
ResourceManager

2A

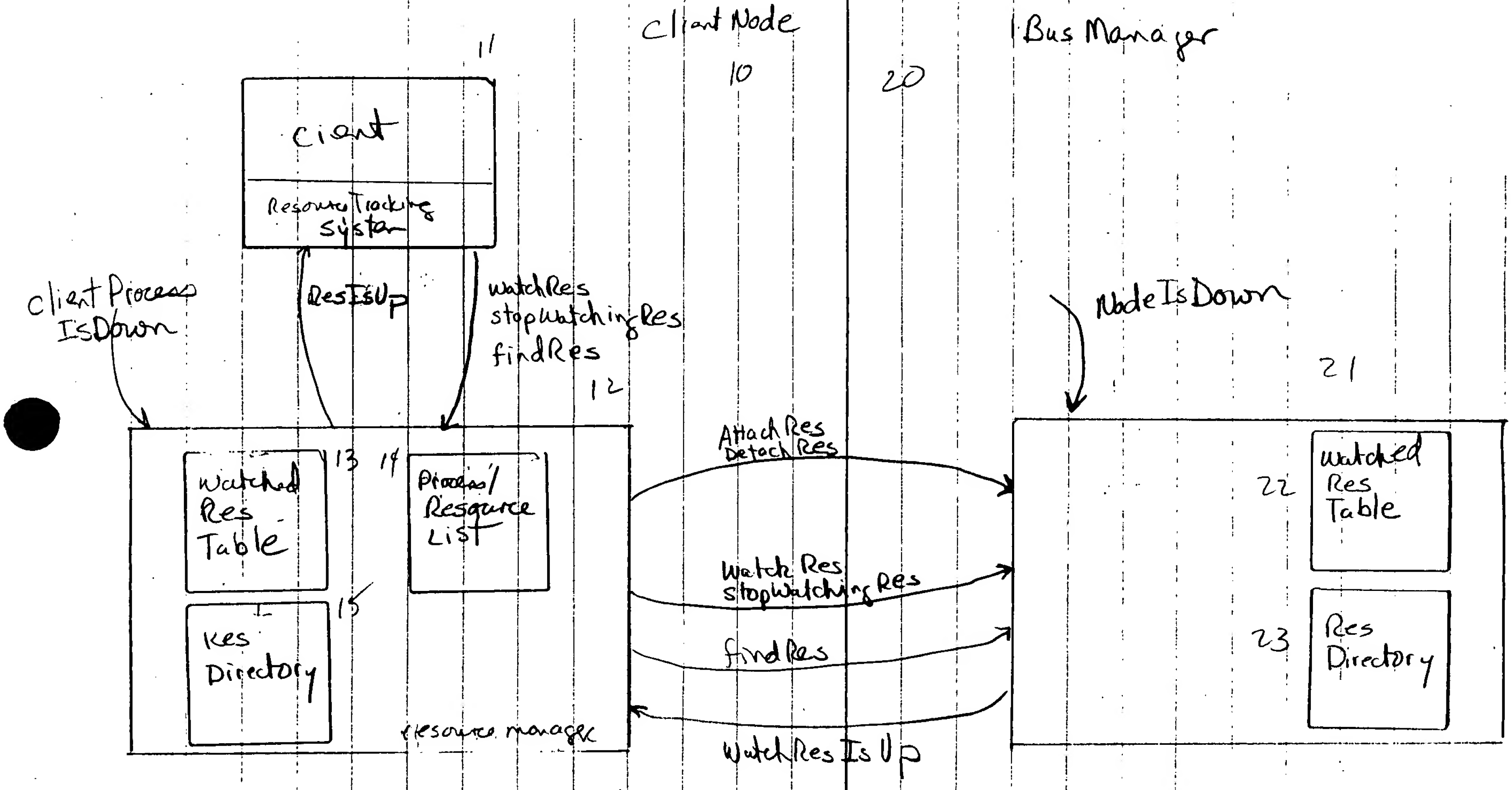

Client Node
Bus Manager
10
20
11
Client
Resource Tracking System
ResIsUp
watchRes
stopWatchingRes
findRes
client Process IsDown
12
13
14
15
Watched Res Table
Process/ Resource List
Res Directory
resource manager
AttachRes
DetachRes
Watch Res
stopWatchingRes
findRes
WatchResIsUp
NodeIsDown
21
22
23
Watched Res Table
Res Directory

Fig 2B

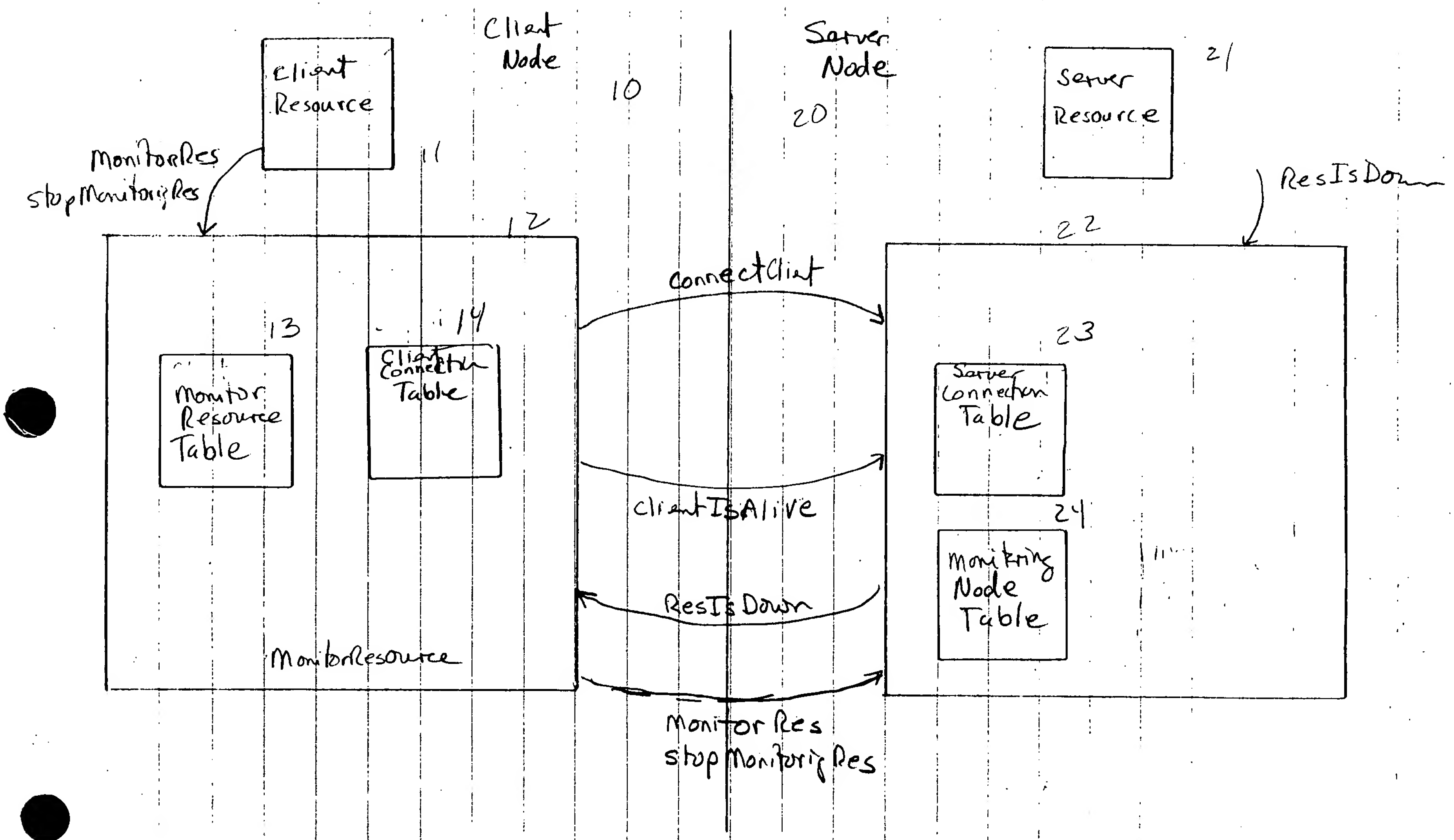
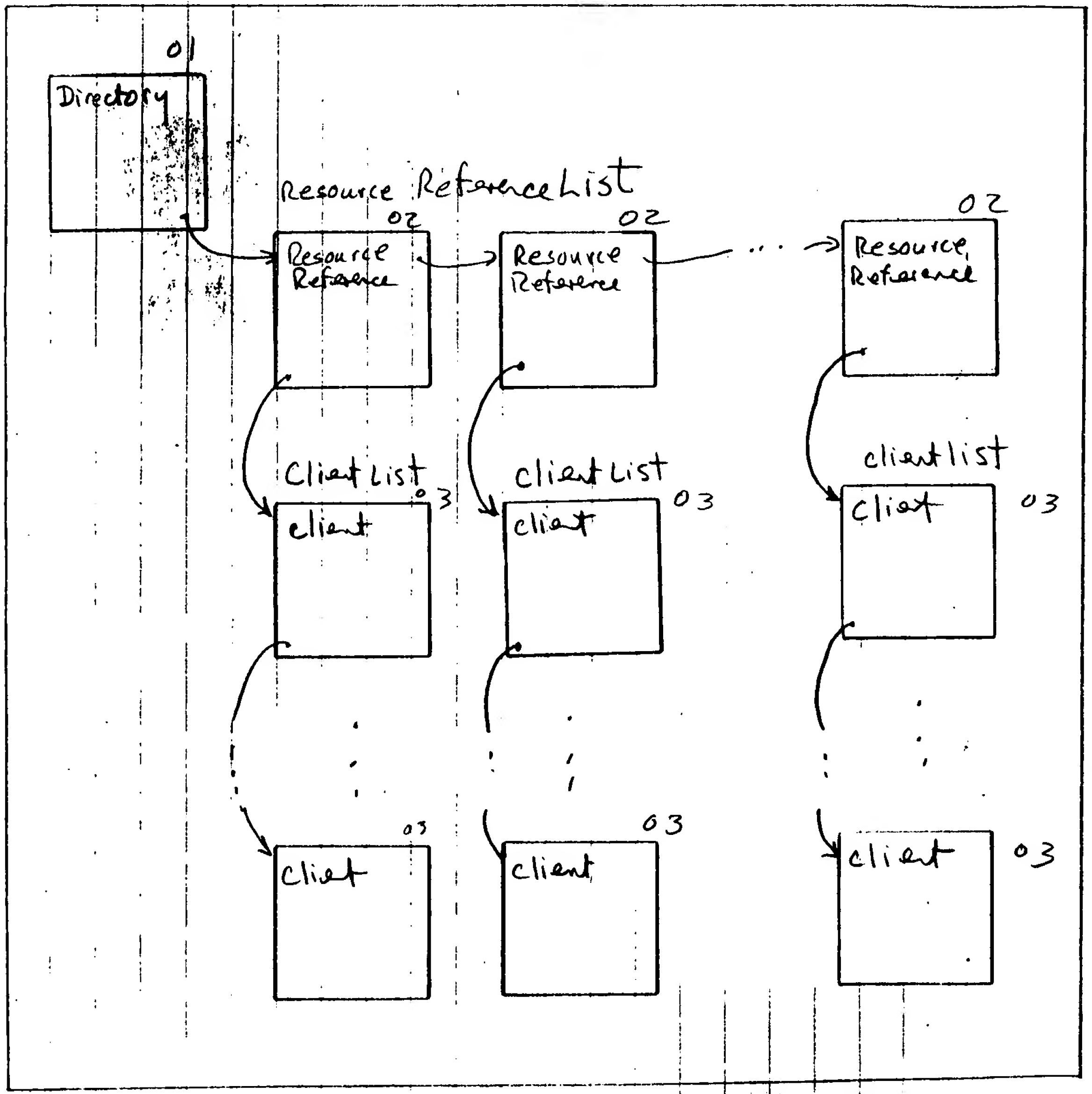

Client Node
10
Server Node
20
Client Resource
11
Server Resource
21
MonitorRes
stopMonitoringRes
ResIsDown
12
22
13
14
Monitor Resource Table
Client Connection Table
MonitorResource
23
24
Server Connection Table
Monitoring Node Table
connectClient
clientIsAlive
ResIsDown
MonitorRes
stopMonitoringRes

Fig 2C


Directory
01
Resource Reference List
02
Resource Reference
02
Resource Reference
02
Resource Reference
Client List
03
client
client List
03
client
clientlist
03
Cliet
03
Cliet
03
client
03
client

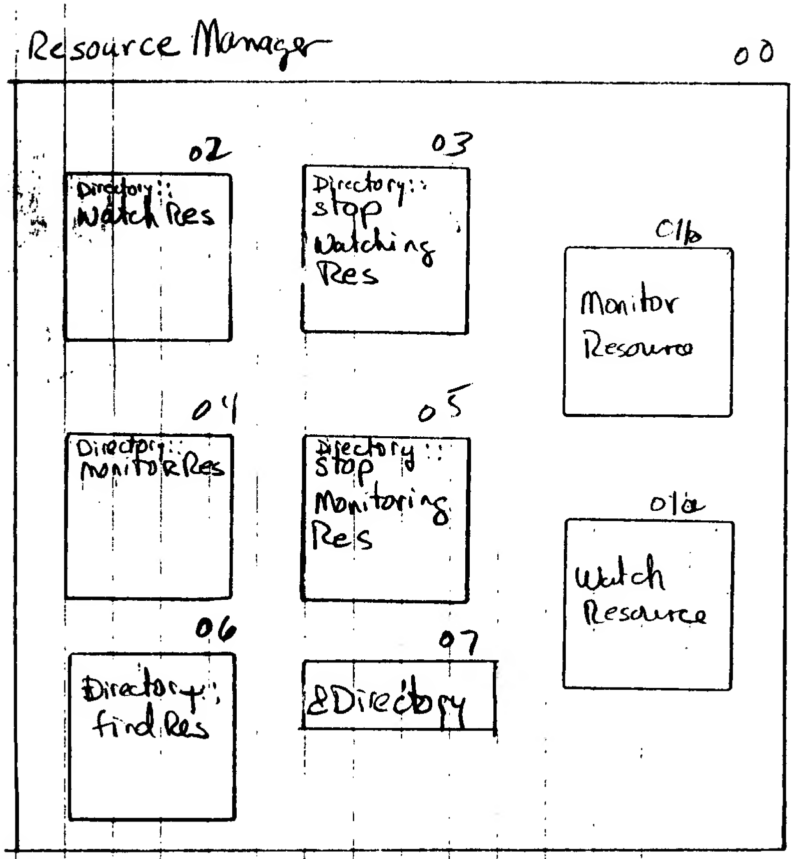

Resource Manager
00
02
Directory::
Watch Res
03
Directory::
stop
Watching
Res
01b
Monitor
Resource
04
Directory::
monitorRes
05
Directory::
stop
Monitoring
Res
01a
Watch
Resource
06
Directory::
find Res
07
&Directory

Fig 4

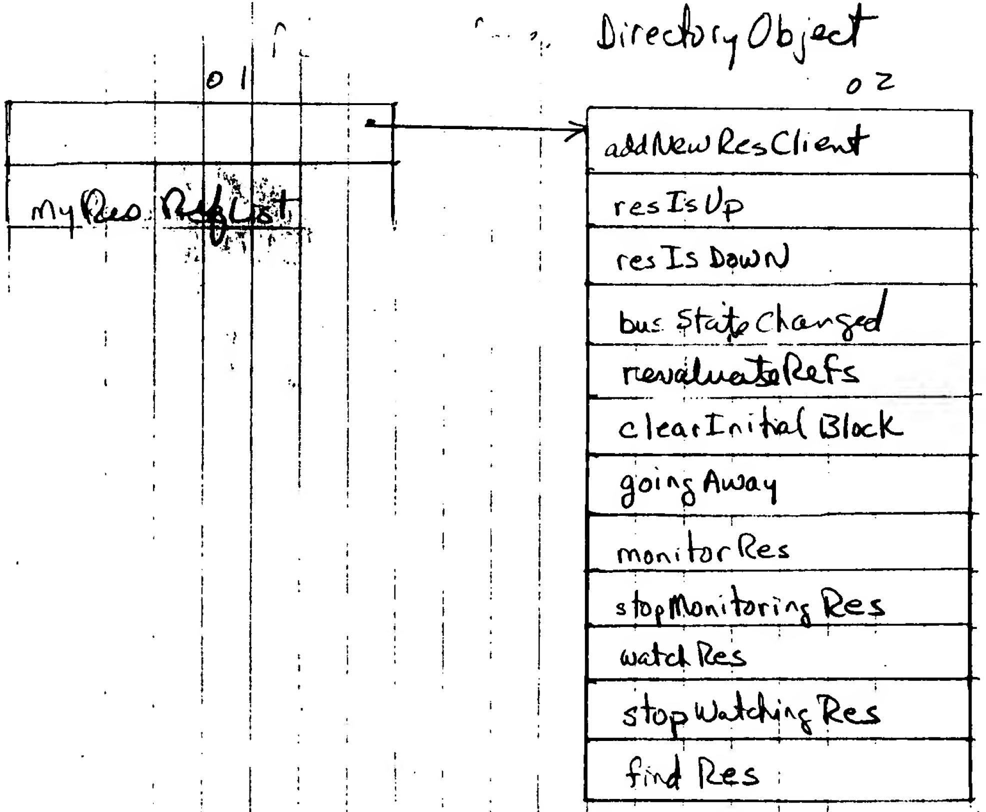

01
myRes RefList
Directory Object
02
addNewResClient
resIsUp
resIsDown
busStateChanged
revaluateRefs
clearInitialBlock
goingAway
monitorRes
stopMonitoringRes
watchRes
stopWatchingRes
findRes

Fig 5

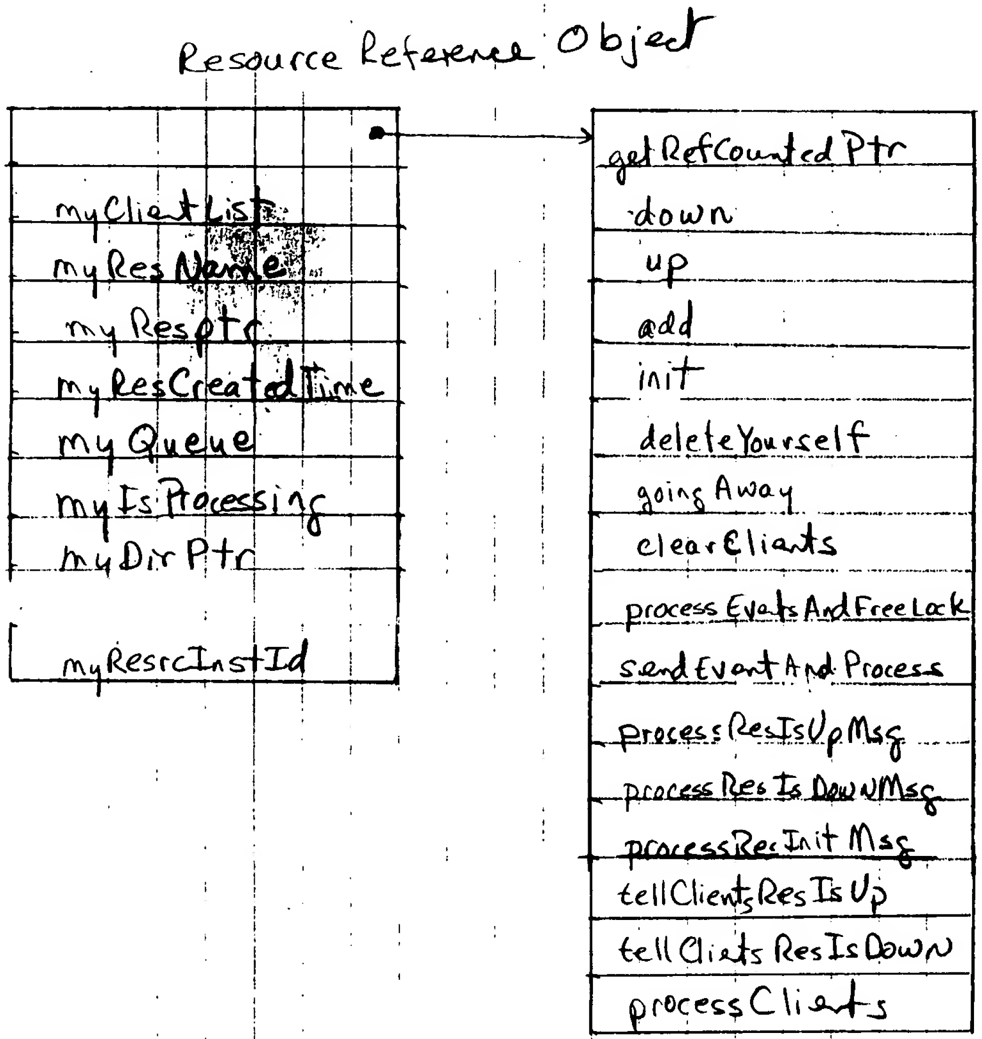

Resource Reference Object
myClientList
myResName
myResptr
myResCreatedTime
myQueue
myIsProcessing
myDirPtr
myResrcInstId
getRefCountedPtr
down
up
add
init
deleteYourself
goingAway
clearClients
processEventsAndFreeLock
sendEventAndProcess
processResIsUpMsg
processResIsDownMsg
processResInitMsg
tellClientsResIsUp
tellClientsResIsDown
processClients

Fig 6

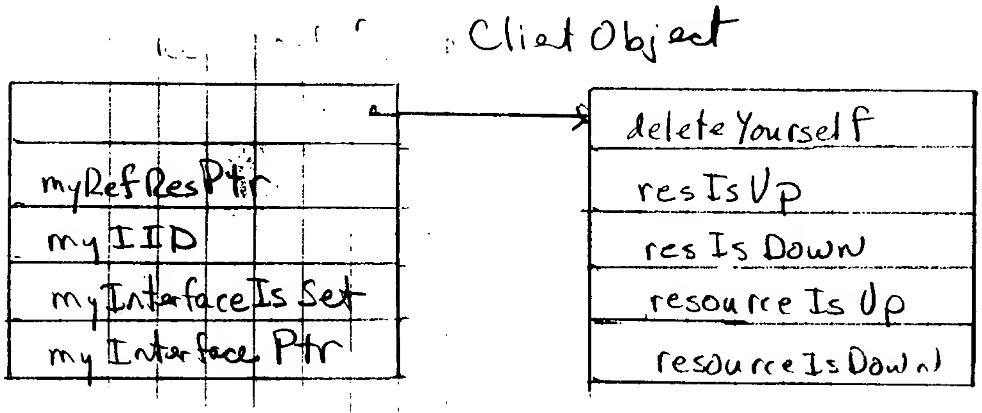

Cliet Object
myRefResPtr
myIID
myInterfaceIsSet
myInterfacePtr
deleteYourself
resIsUp
resIsDown
resourceIsUp
resourceIsDown

Fig 7

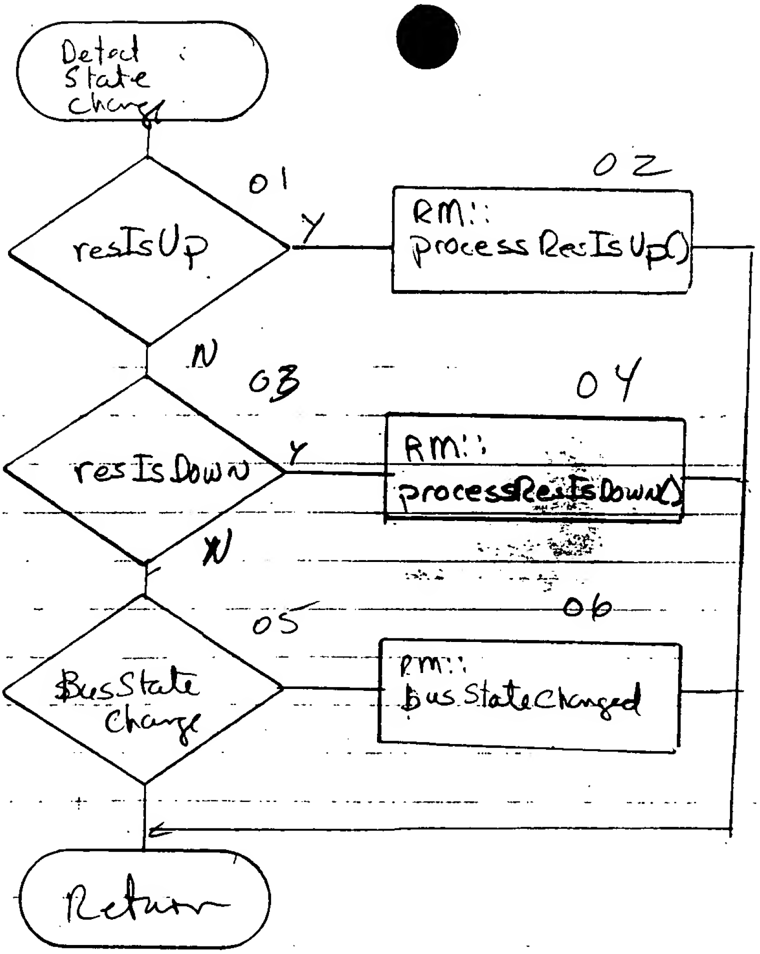
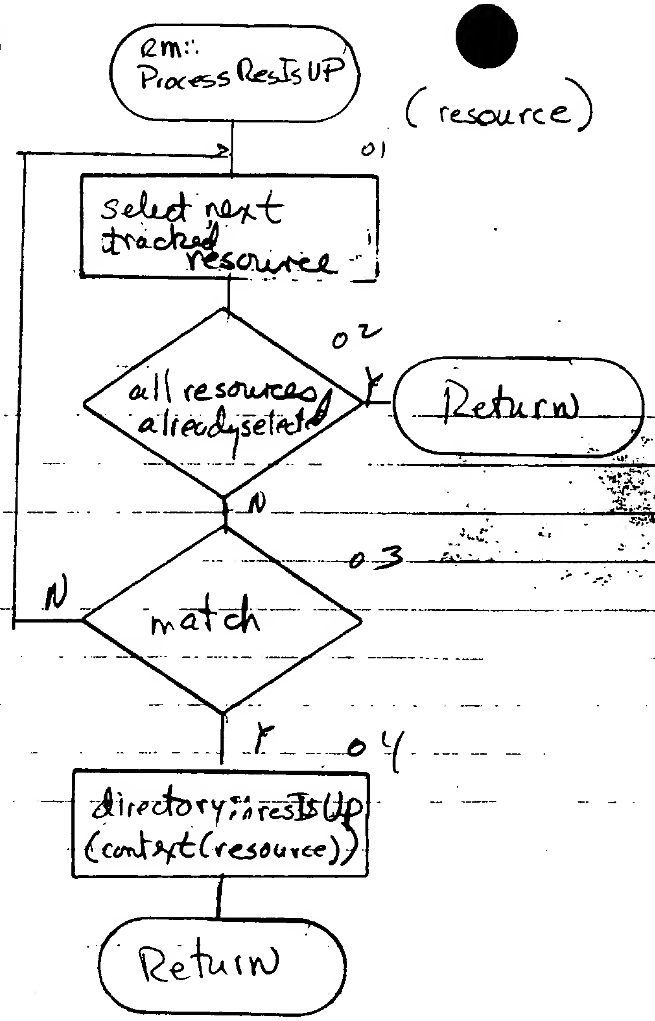

Detect State Change
01
resIsUp
Y
02
RM:: process RecIsUp()
N
03
resIsDown
Y
04
RM:: processResIsDown()
N
05
BusState Change
06
RM:: BusStateChanged
Return

Fig 8



Rm::
ProcessResIsUP
(resource)
01
select next tracked resource
02
all resources already selected
Y
Return
N
03
N
match
Y
04
directory::resIsUp (context(resource))
Return

Fig 9

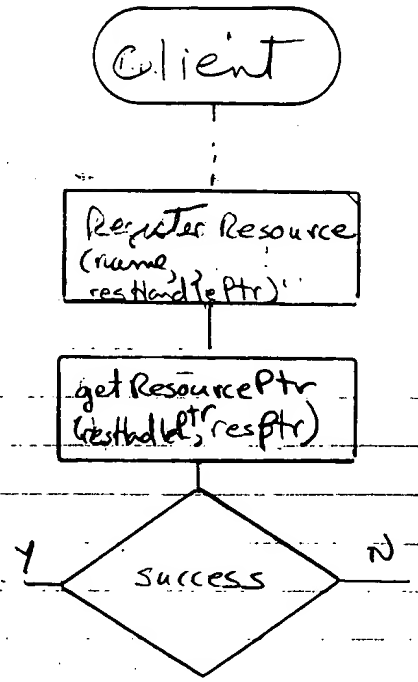

Client
RegisterResource (name, resHandlePtr)
getResourcePtr (resHandlePtr, resPtr)
success
Y
N

Fig 10

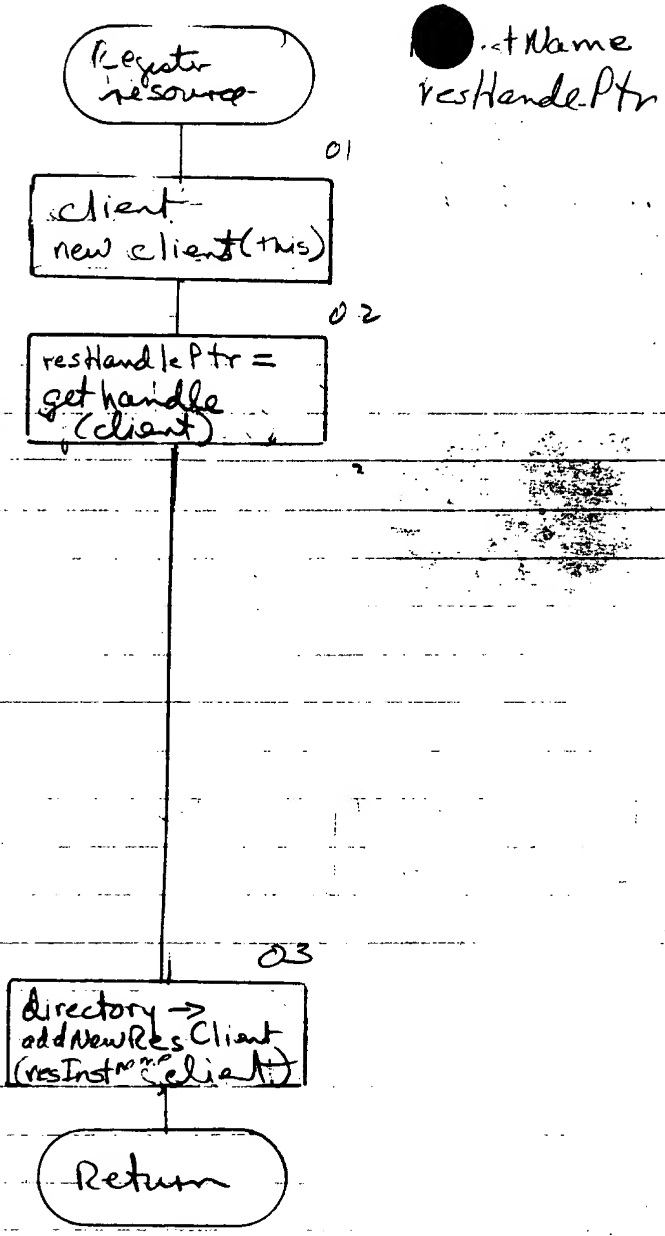

tName
resHandlePtr
Register resource
01
client
new client(this)
02
resHandlePtr =
gethandle
(client)
03
directory ->
addNewResClient
(resInstName, client)
Return

Figure 11

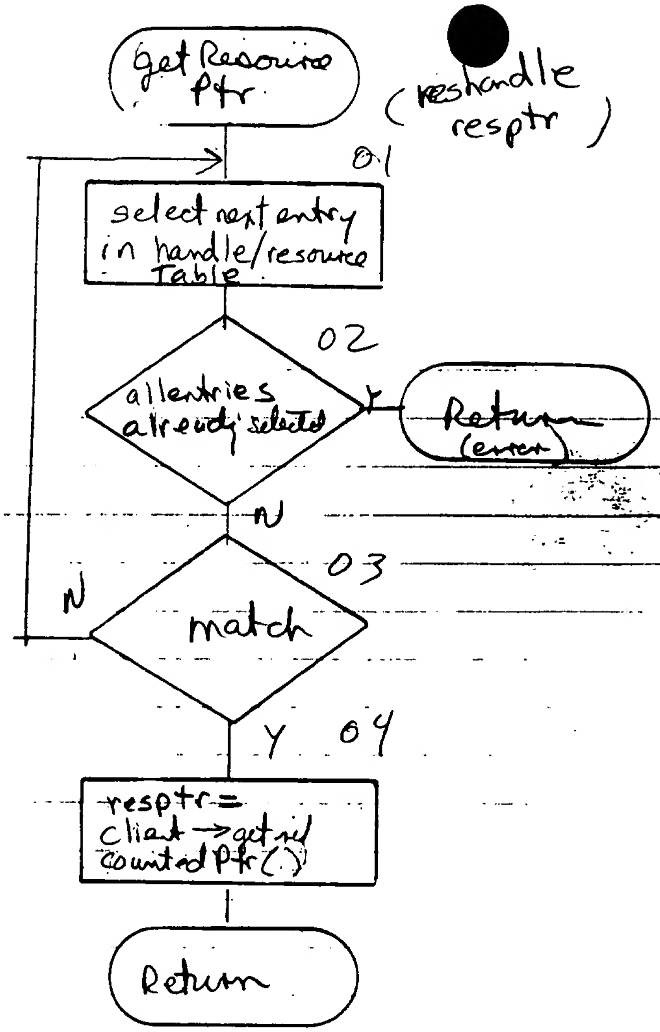

Get Resource Ptr
(reshandle resptr)
01
select next entry in handle/resource Table
02
all entries already selected
Y
Return (error)
N
03
N
match
Y
04
resptr = client→getrid countedPtr( )
Return

Fig 12

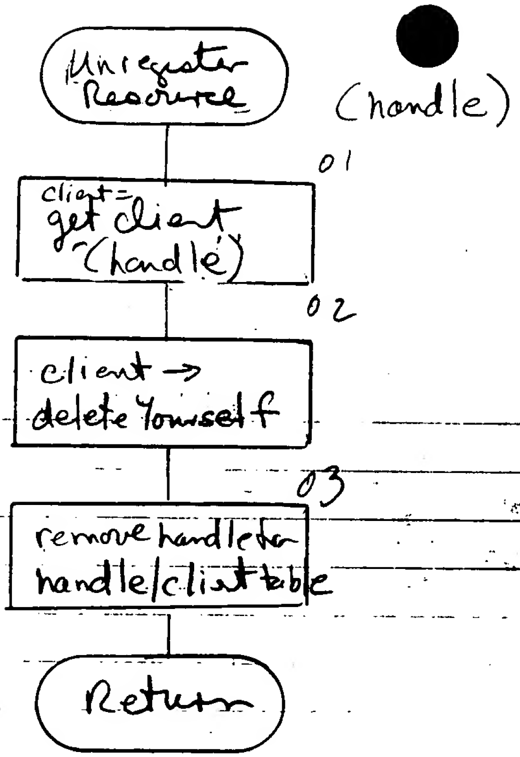

Unregister Resource
(handle)
01
client = get client (handle)
02
client → deleteYourself
03
remove handle for handle/client table
Return

Fig 13

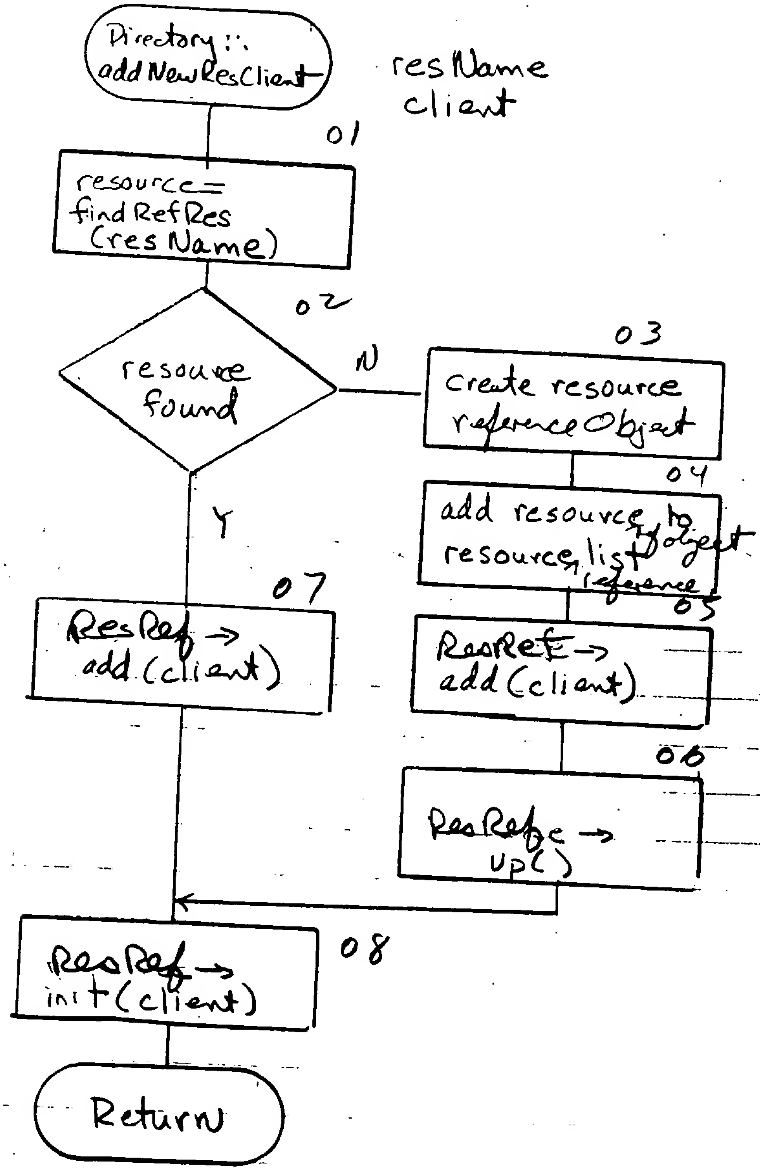

Directory::
addNewResClient
resName
client
01
resource=
findRefRes
(resName)
02
resource
found
N
03
create resource
referenceObject
04
add resource to
resource list
object
reference
05
ResRef→
add(client)
06
ResRefe→
up()
Y
07
ResRef→
add(client)
08
ResRef→
init(client)
Return

Fig 14

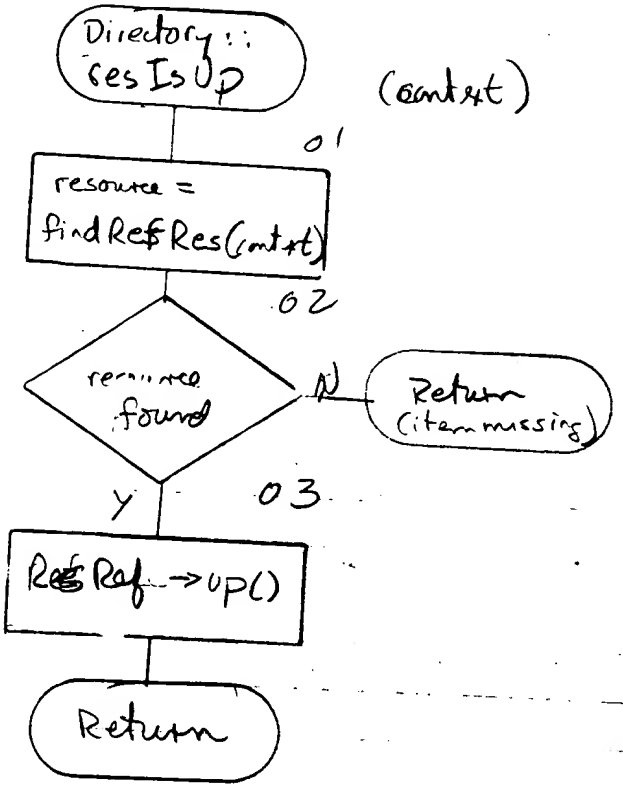

Directory::
resIsUp
(context)
01
resource =
findRefRes(context)
02
resource
found
N
Return
(item missing)
Y
03
~~Res~~ Ref->up()
Return

Fig 15

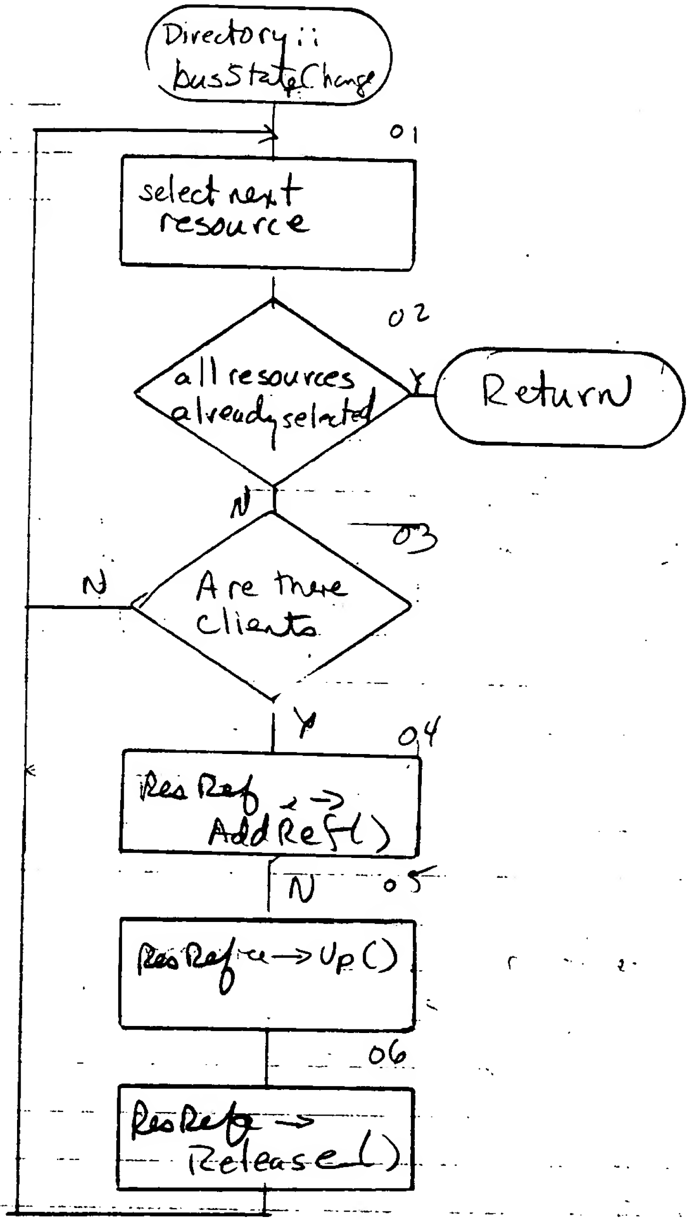

Directory::
busStateChange
01
select next resource
02
all resources already selected
Y
Return
N
03
N
Are there clients
Y
04
ResRef->
AddRef()
N
05
ResRef->Up()
06
ResRef->
Release()

Fig. 16

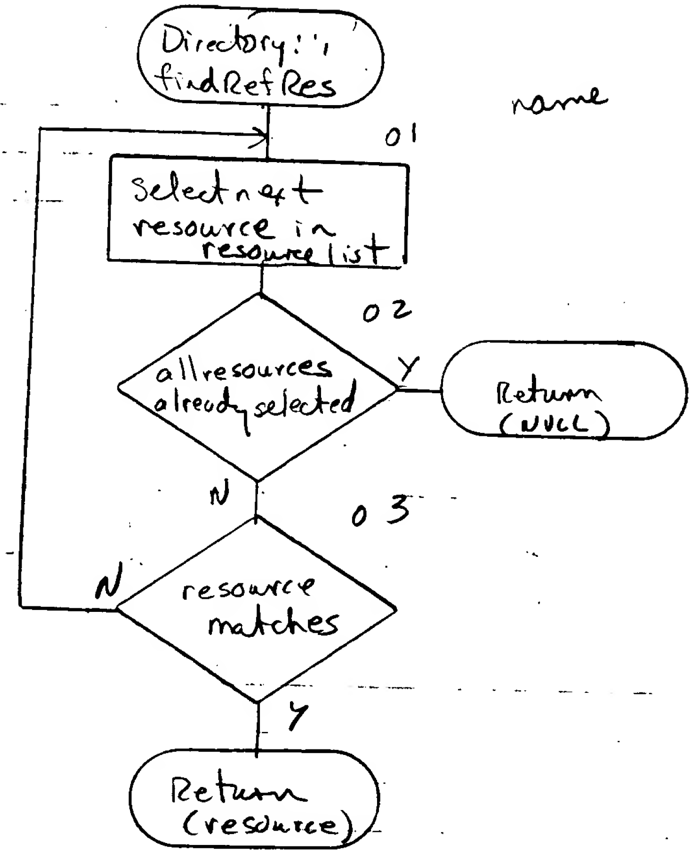

Directory::
findRefRes
name
01
select next
resource in
resource list
02
all resources
already selected
Y
Return
(NULL)
N
03
N
resource
matches
Y
Return
(resource)

Fig 17

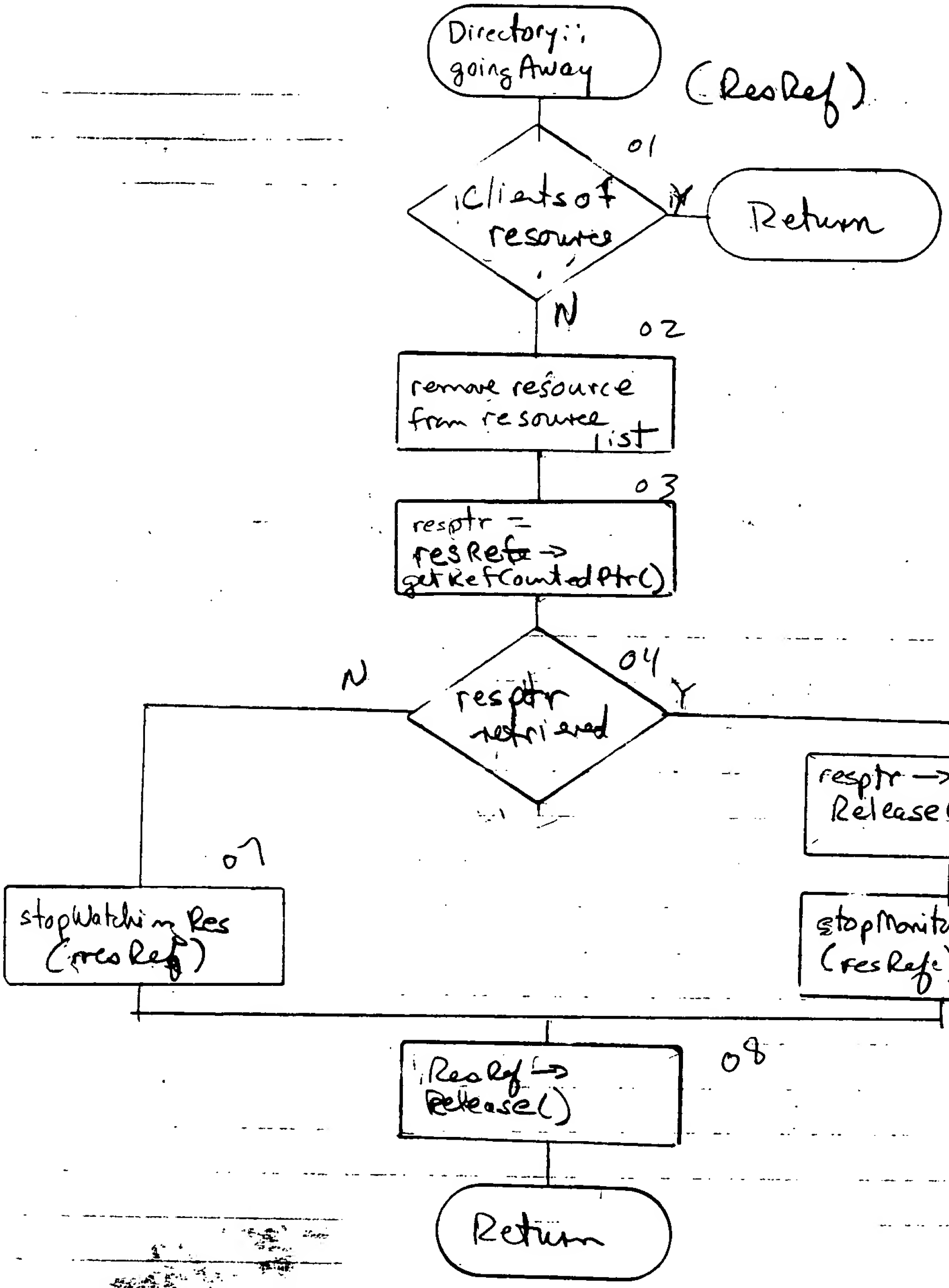

Directory::
goingAway
(ResRef)
01
Clients of
resource
Y
Return
N
02
remove resource
from resource list
03
resptr =
resRef ->
getRefCountedPtr()
04
N
resptr
retrieved
Y
05
resptr ->
Release()
06
stopMonitoringRes
(resRef)
07
stopWatchingRes
(resRef)
08
ResRef ->
Release()
Return

Figure 18

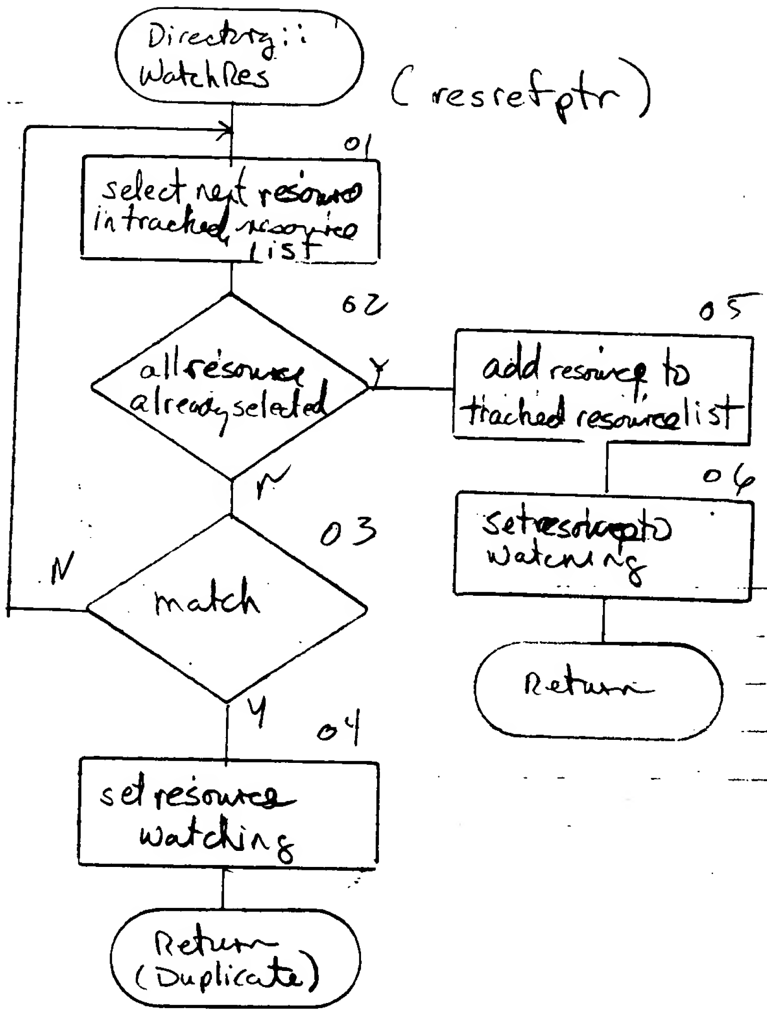

Directory::
WatchRes
(resrefptr)
01
select next resource in tracked resource list
02
all resource already selected
Y
N
05
add resource to tracked resource list
06
set resource to watching
Return
03
N
match
Y
04
set resource watching
Return (Duplicate)

Fig 19

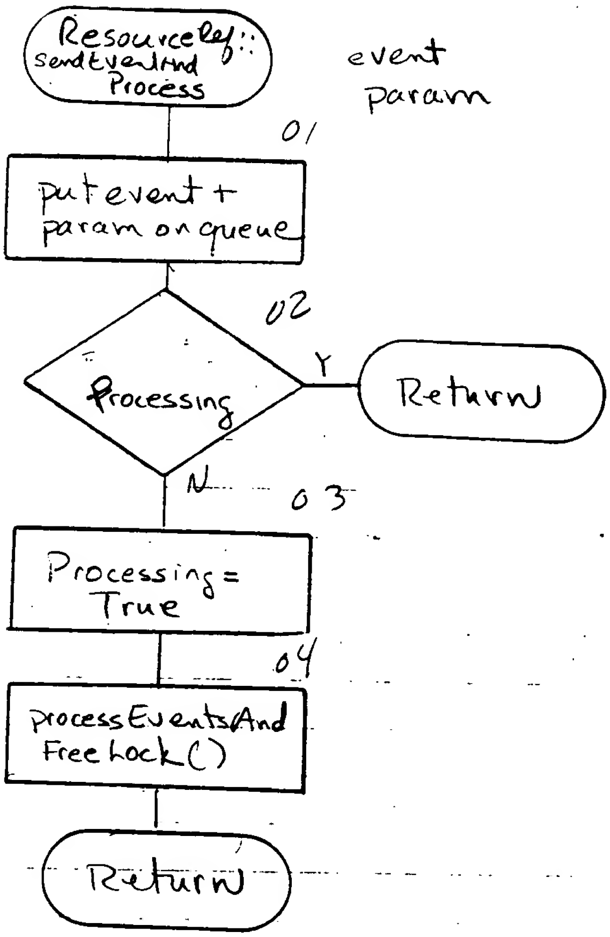

ResourceRef:: sendEventAnd Process
event
param
01
put event + param on queue
02
Processing
Y
Return
N
03
Processing = True
04
processEventsAnd FreeLock()
Return

Fig 20

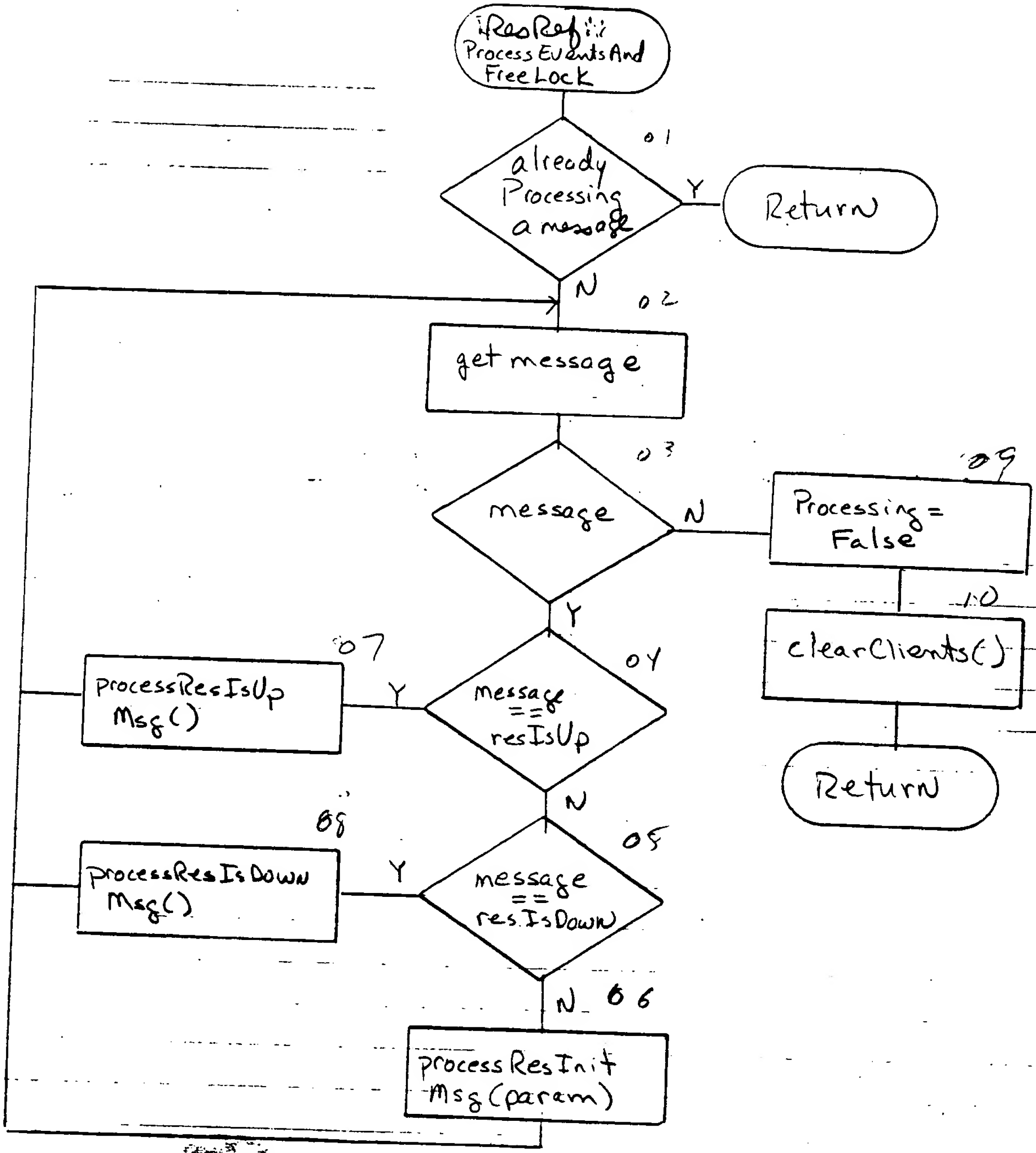

iResRef::
Process Events And
Free Lock
01
already
Processing
a message
Y
Return
N
02
get message
03
message
N
09
Processing =
False
10
clearClients()
Return
Y
04
message
==
resIsUp
Y
07
processResIsUp
Msg()
N
05
message
==
res IsDown
Y
08
processResIsDown
Msg()
N
06
processResInit
Msg(param)

Fig 21

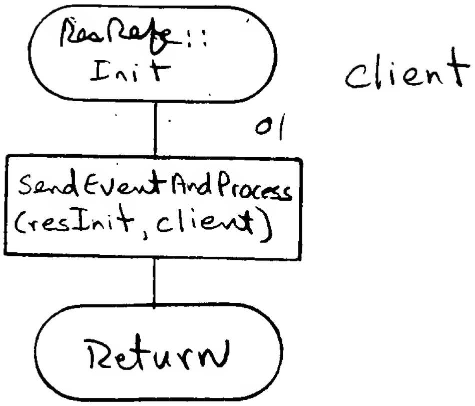

ResRefe::
Init
client
01
SendEventAndProcess
(resInit, client)
Return

Fig 22

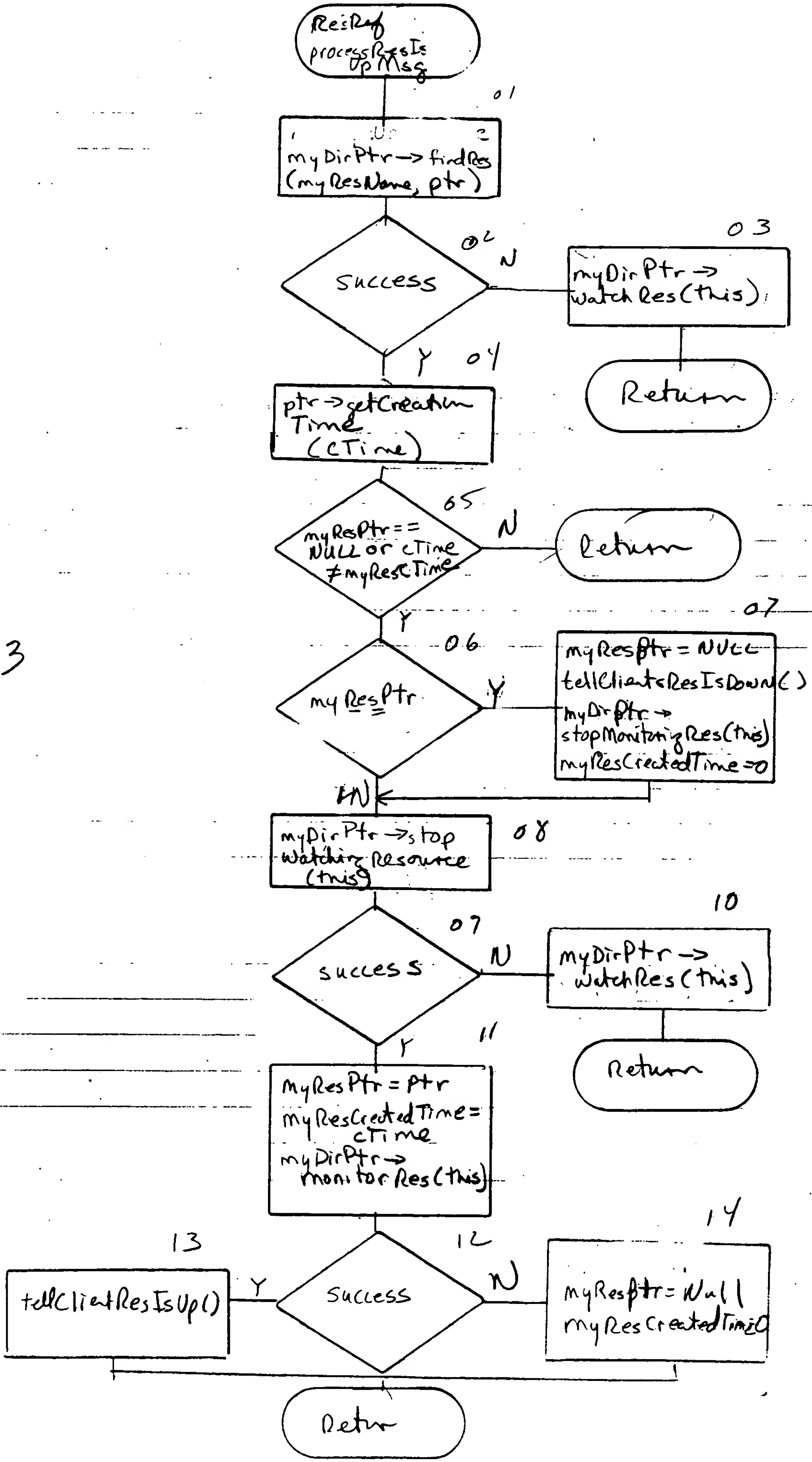
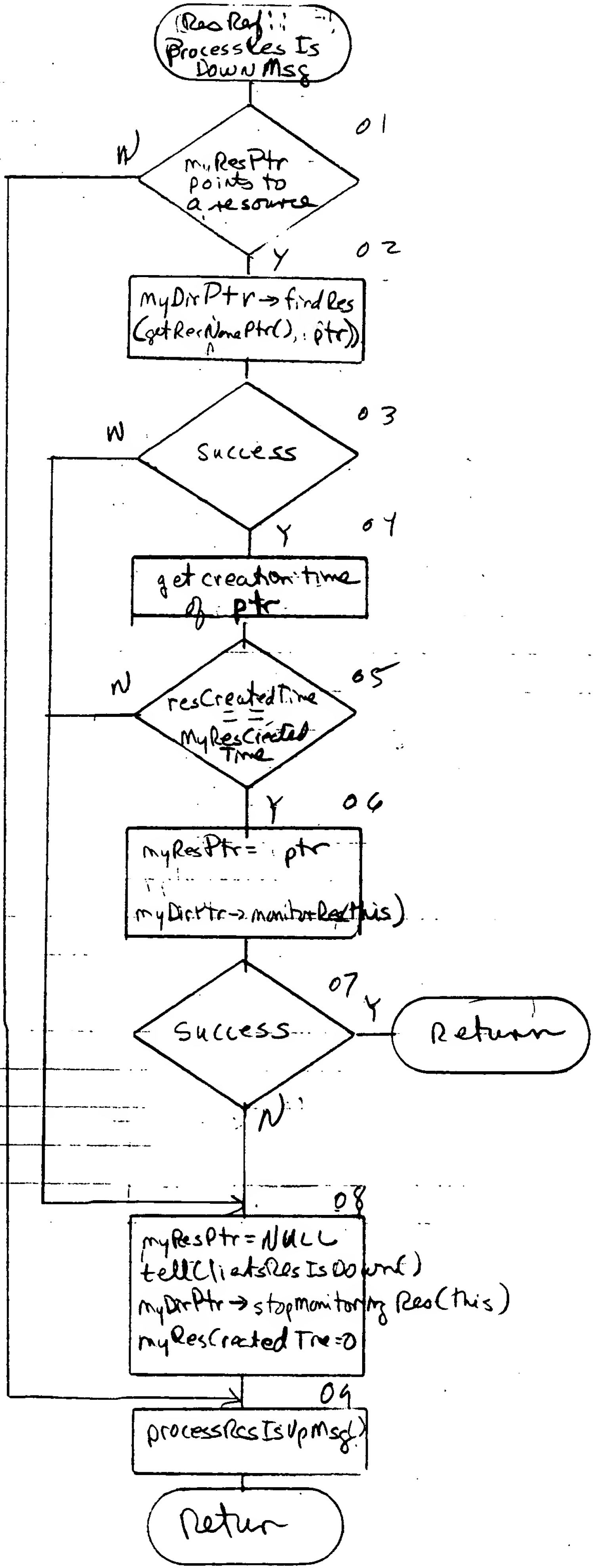

ResRef processResIsUpMsg
01
myDirPtr -> findRes (myResName, ptr)
02
success
N
03
myDirPtr -> watchRes(this)
Return
Y
04
ptr -> getCreationTime (cTime)
05
myResPtr == NULL or cTime ≠ myResCTime
N
Return
Y
06
myResPtr
Y
07
myResPtr = NULL
tellClientResIsDown()
myDirPtr -> stopMonitoringRes(this)
myResCreatedTime = 0
N
myDirPtr -> stop watching Resource (this)
08
09
success
N
10
myDirPtr -> watchRes(this)
Return
Y
11
myResPtr = ptr
myResCreatedTime = cTime
myDirPtr -> monitorRes(this)
12
success
Y
13
tellClientResIsUp()
N
14
myResPtr = Null
myResCreatedTime=0
Return

Fig 23

Fig 24


ResRef::
ProcessResIsDownMsg
01
N
myResPtr points to a resource
Y
02
myDirPtr→findRes (getResNamePtr(), ptr))
03
N
Success
Y
04
get creation time of ptr
05
N
resCreatedTime == MyResCreatedTime
Y
06
myResPtr = ptr
myDirPtr→monitorRes(this)
07
Y
Success
Return
N
08
myResPtr=NULL
tellClientsResIsDown()
myDirPtr→stopMonitoringRes(this)
myResCreatedTime=0
09
processResIsUpMsg()
Return

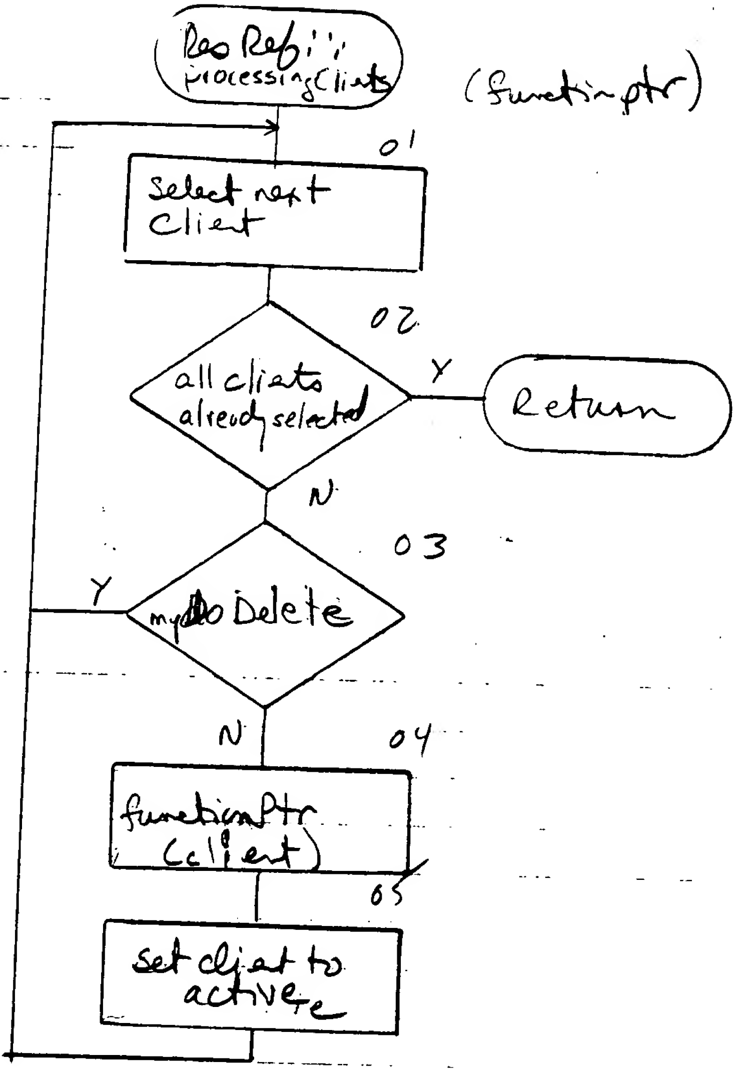

ResRef::processingClients
(functionptr)
01
Select next Client
02
all clients already selected
Y
Return
N
03
Y
myDoDelete
N
04
functionPtr (client)
05
set client to active

Fig 25

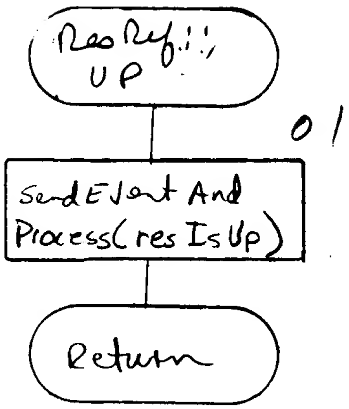


Fig 24

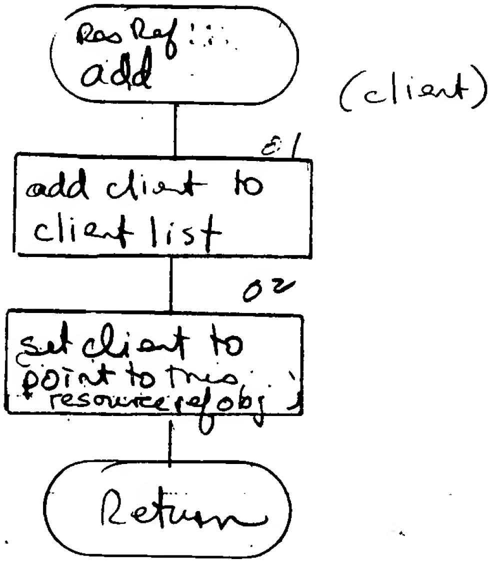

Res Ref:::
add
(client)
01
add client to client list
02
set client to point to this resource ref obj
Return

Fig 27

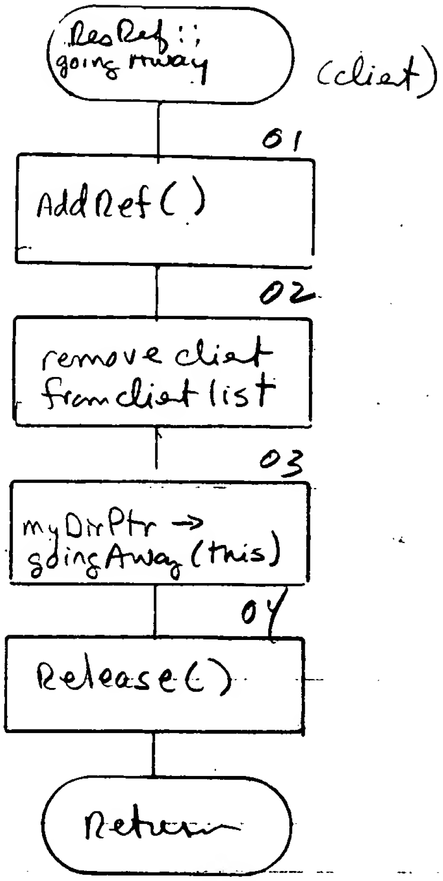

ResRef::
going Away
(cliat)
01
AddRef()
02
remove cliat
from cliat list
03
myDirPtr →
goingAway(this)
04
Release()
Return

Fig 28

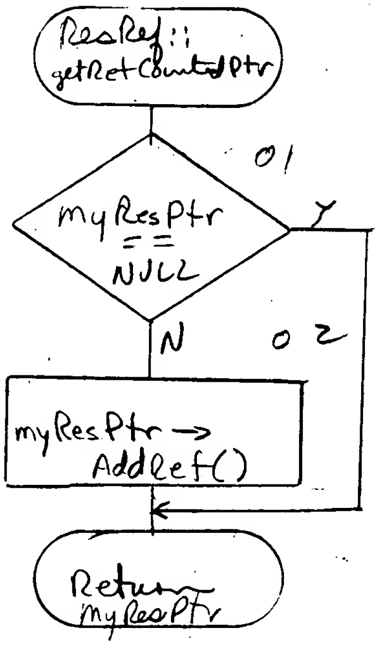

ResRef::
getRefCountedPtr
01
myResPtr
==
NULL
Y
N
02
myResPtr ->
AddRef()
Return
myResPtr

Fig 29

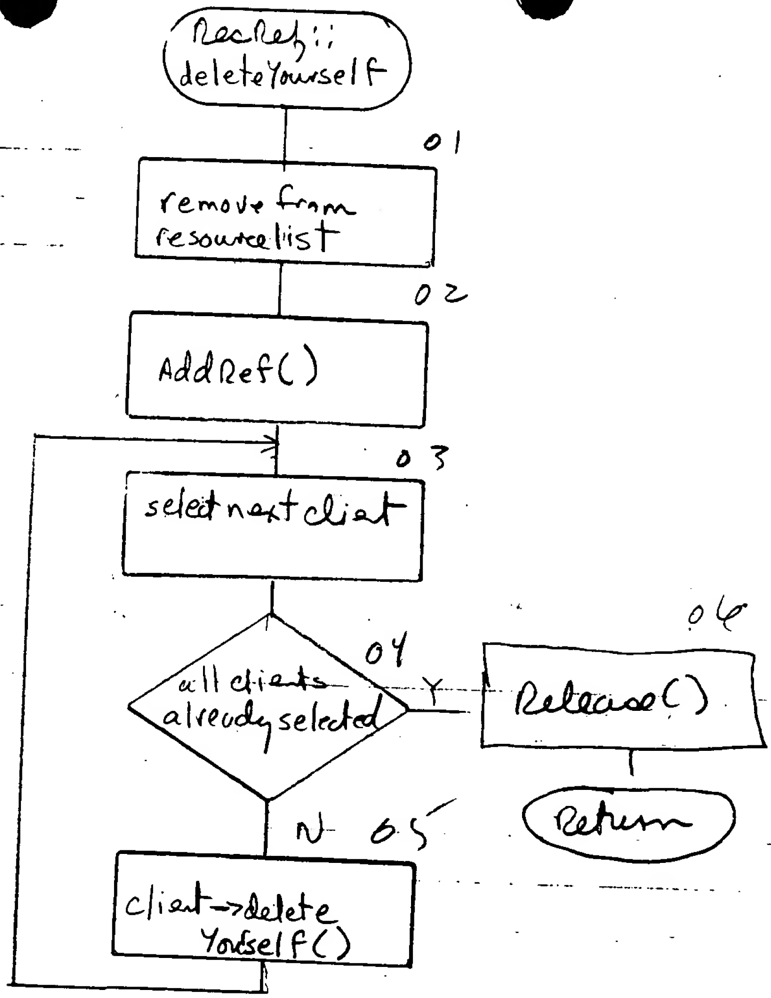

RecRef::
deleteYourself
01
remove from
resourcelist
02
AddRef()
03
select next cliet
04
all clients
already selected
Y
06
Release()
Return
N
05
client->delete
Yorself()

Fig 30

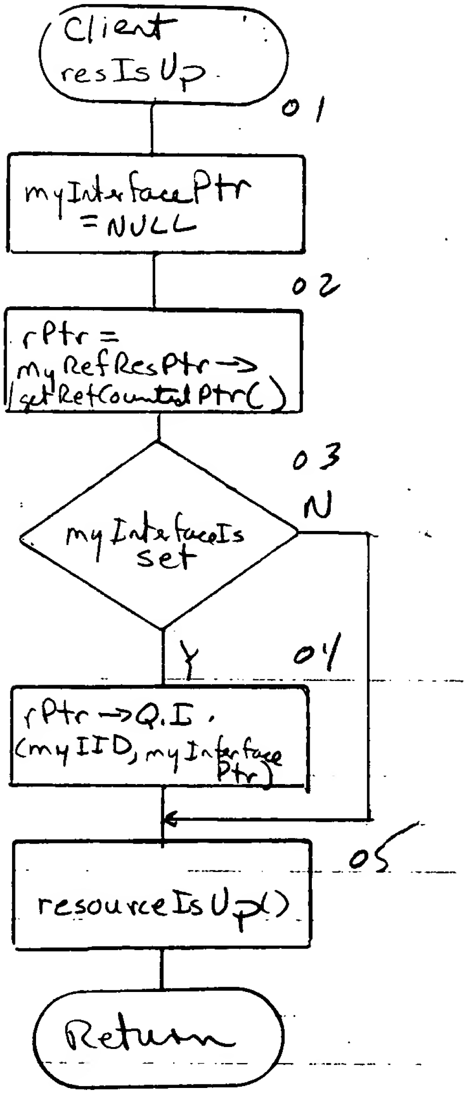

Client resIsUp
01
myInterfacePtr =NULL
02
rPtr = myRefResPtr-> getRefCountedPtr()
03
myInterfaceIs set
N
Y
04
rPtr->Q.I. (myIID, myInterfacePtr)
05
resourceIsUp()
Return

Fig 31

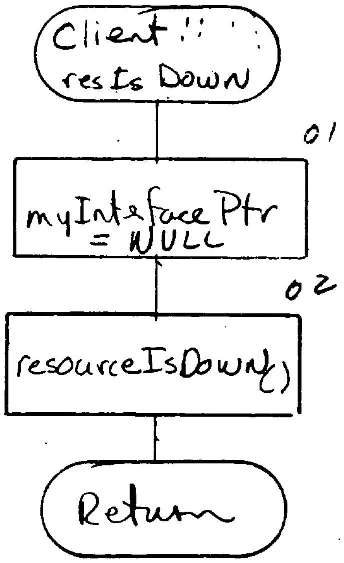

Client :: resIsDown
01
myInteface Ptr = NULL
02
resourceIsDown()
Return

Fig 32

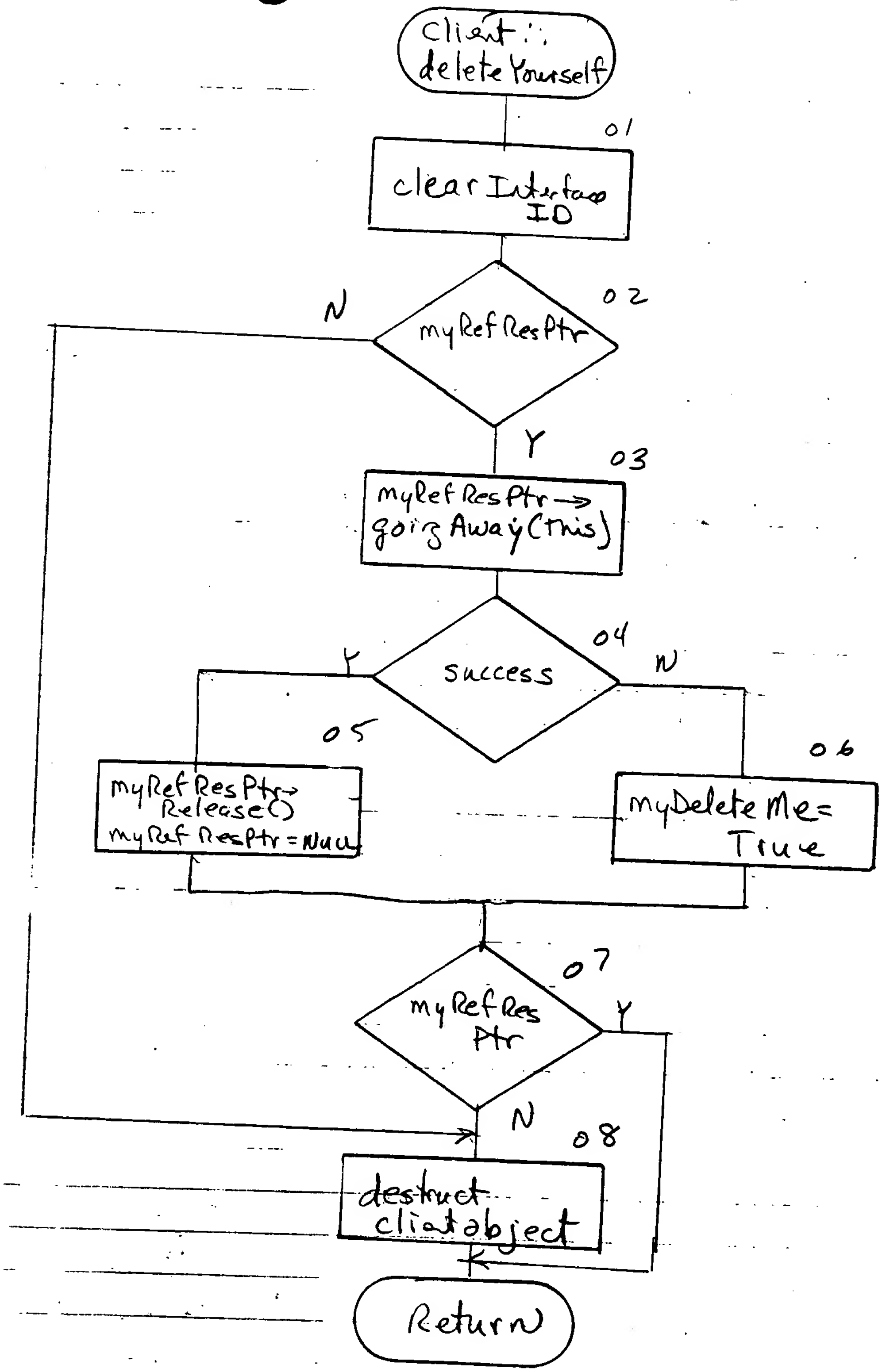
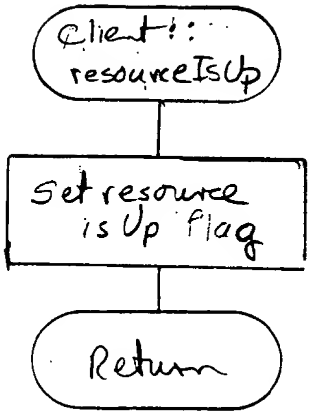

Client :: deleteYourself
01
clear Interface ID
02
myRefResPtr
N
Y
03
myRefResPtr→ goingAway(This)
04
success
Y
N
05
myRefResPtr→ Release()
myRefResPtr=Null
06
myDeleteMe= True
07
myRefRes Ptr
Y
N
08
destruct client object
Return

Figure 33


Client i::
resourceIsUp
Set resource
isUp flag
Return

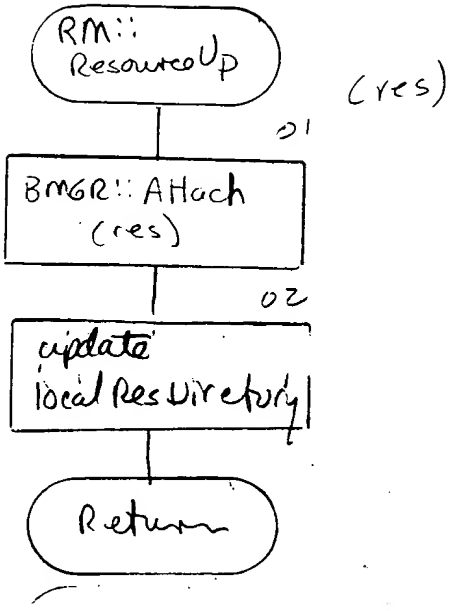

RM::
ResourceUp
(res)
01
BMGR:: Attach
(res)
02
update
local ResDirectory
Return

Fig 35

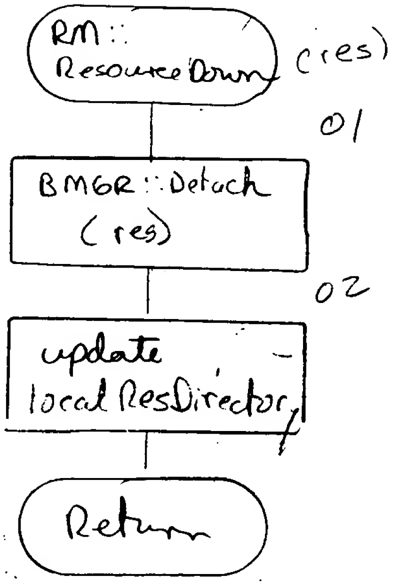

RM::
ResourceDown
(res)
01
BMGR::Detach
(res)
02
update
localResDirectory
Return

Fig 36

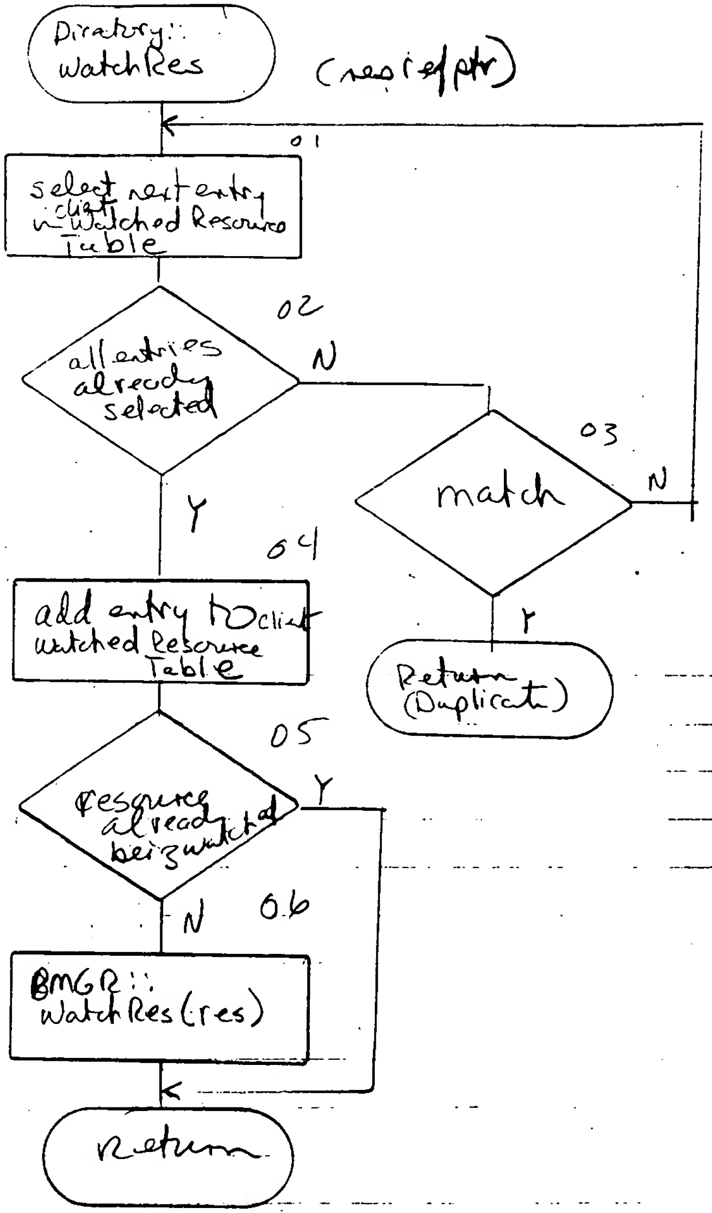

Directory::
WatchRes
(res, ~~self~~ ptr)
01
select next entry in client Watched Resource Table
02
all entries already selected
N
Y
03
match
N
Y
Return (Duplicate)
04
add entry to client watched Resource Table
05
Resource already being watched
Y
N
06
BMGR::
WatchRes (res)
Return

Fig 37

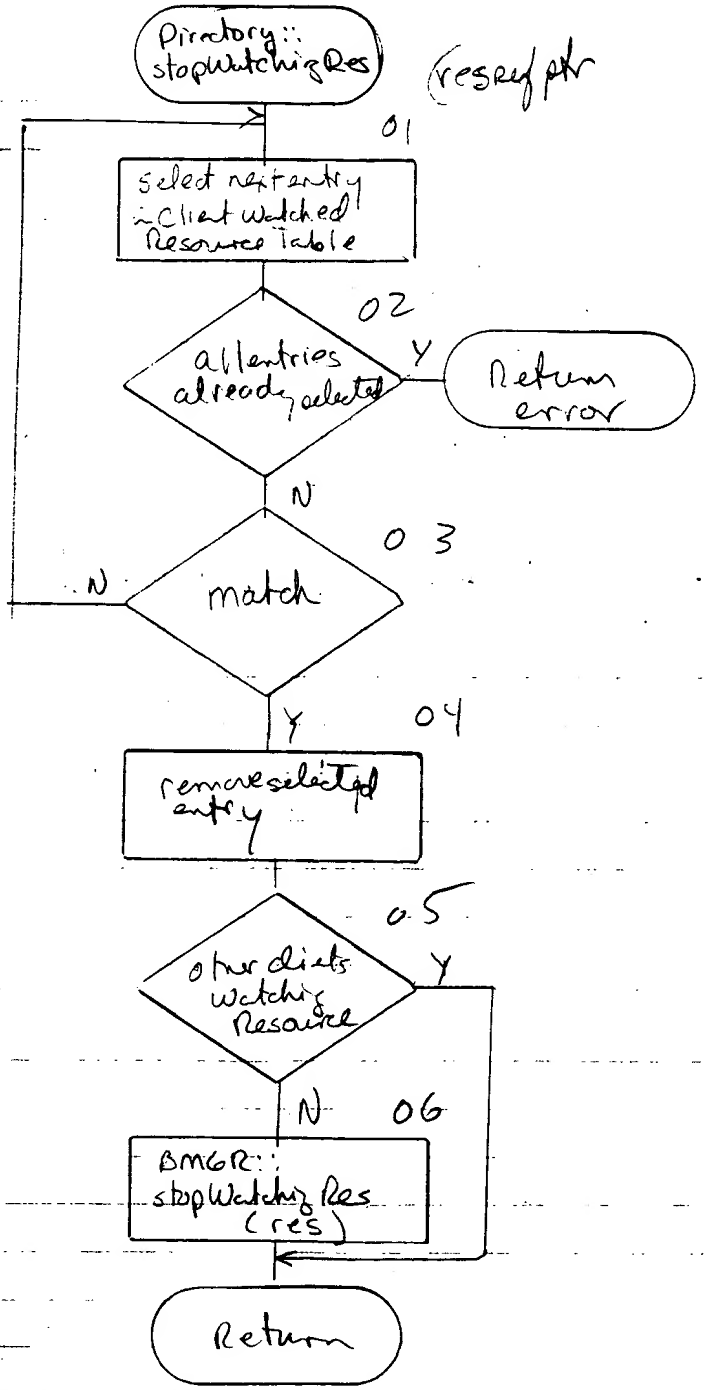

Directory::
stopWatchingRes
(resref ptr
01
select next entry
in Client Watched
Resource Table
02
all entries
already selected
Y
Return
error
N
03
N
match
Y
04
remove selected
entry
05
other clients
watching
Resource
Y
N
06
BMGR::
stopWatching Res
(res)
Return

Fig 38

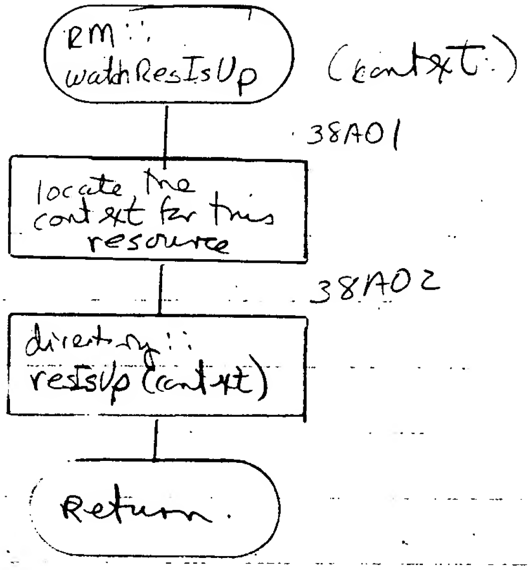

RM::
watchResIsUp
(context:)
38A01
locate the context for this resource
38A02
directory::
resIsUp (context)
Return.

Figure 38A

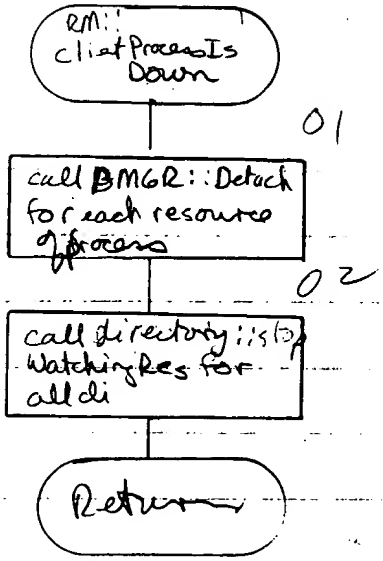
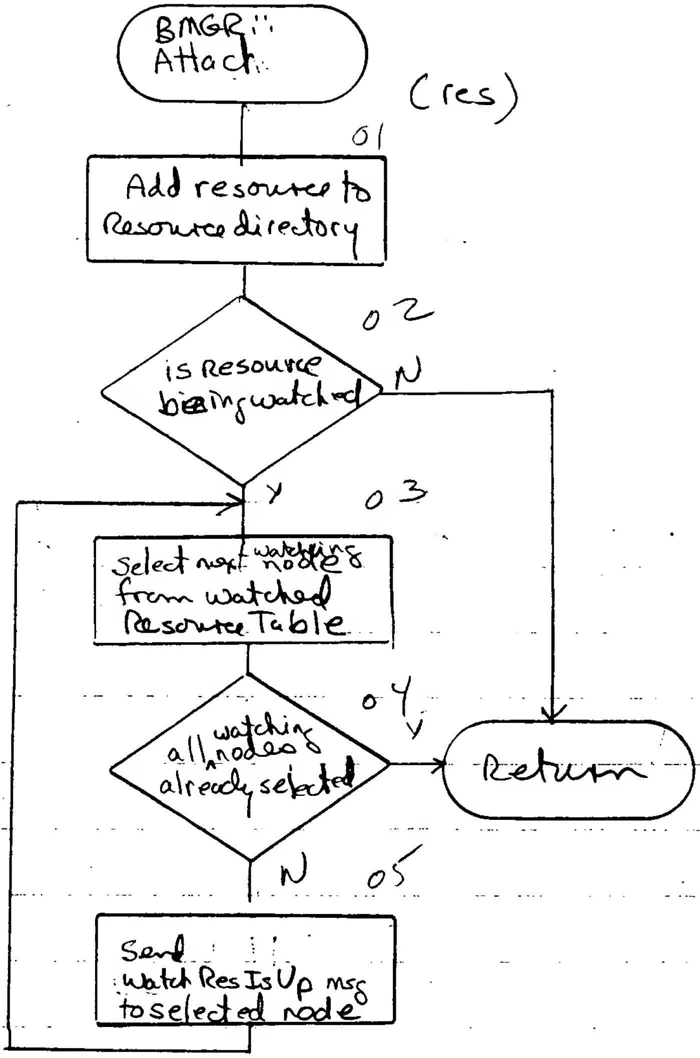

RM::
clientProcessIs
Down
01
call BMGR::Detach
for each resource
of process
02
call directory::stop
WatchingRes for
all di
Return

Fig 39


BMGR::
Attach::
(res)
01
Add resource to Resource directory
02
is Resource being watched
N
Y
03
Select next watching node from watched Resource Table
04
Y
watching all nodes already selected
Return
N
05
Send WatchResIsUp msg to selected node

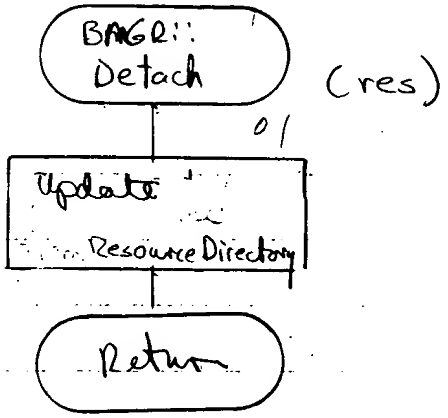


Fig 41

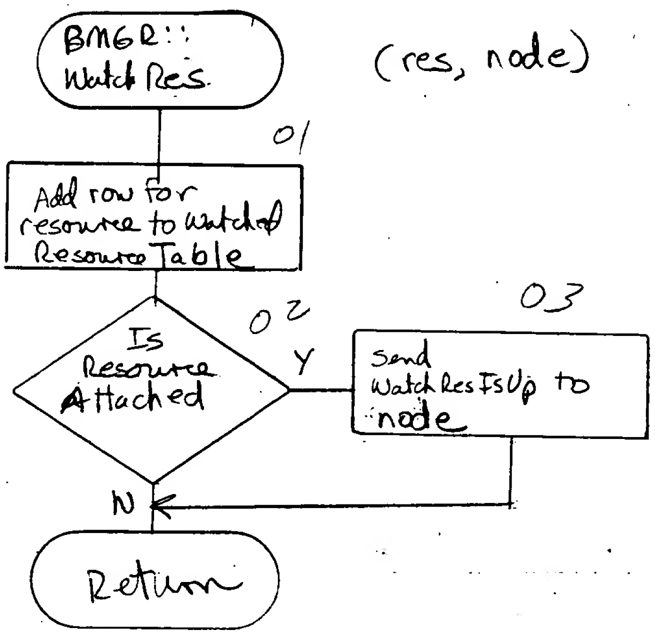

BMGR::
WatchRes
(res, node)
01
Add row for resource to Watched Resource Table
02
Is Resource Attached
Y
03
Send WatchResIsUp to node
N
Return

Fig 42

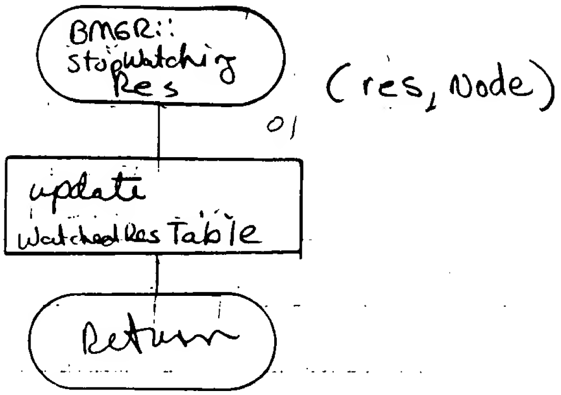
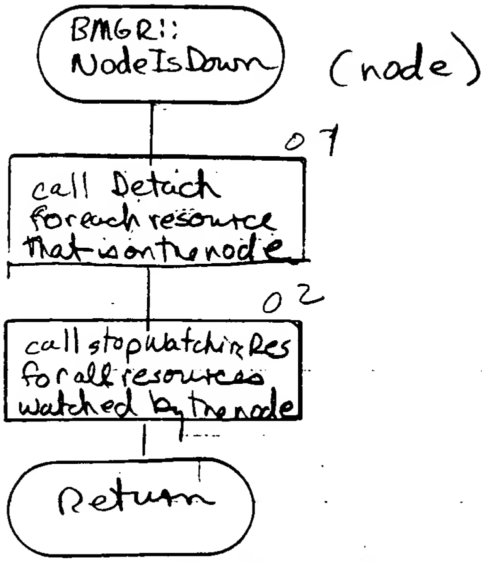

BMGR:: stopWatching Res
(res, Node)
01
update WatchedResTable
Return

Fig 43


BMGR::
NodeIsDown
(node)
01
call Detach
foreach resource
that is on the node
02
call stopWatchingRes
for all resources
watched by the node
Return

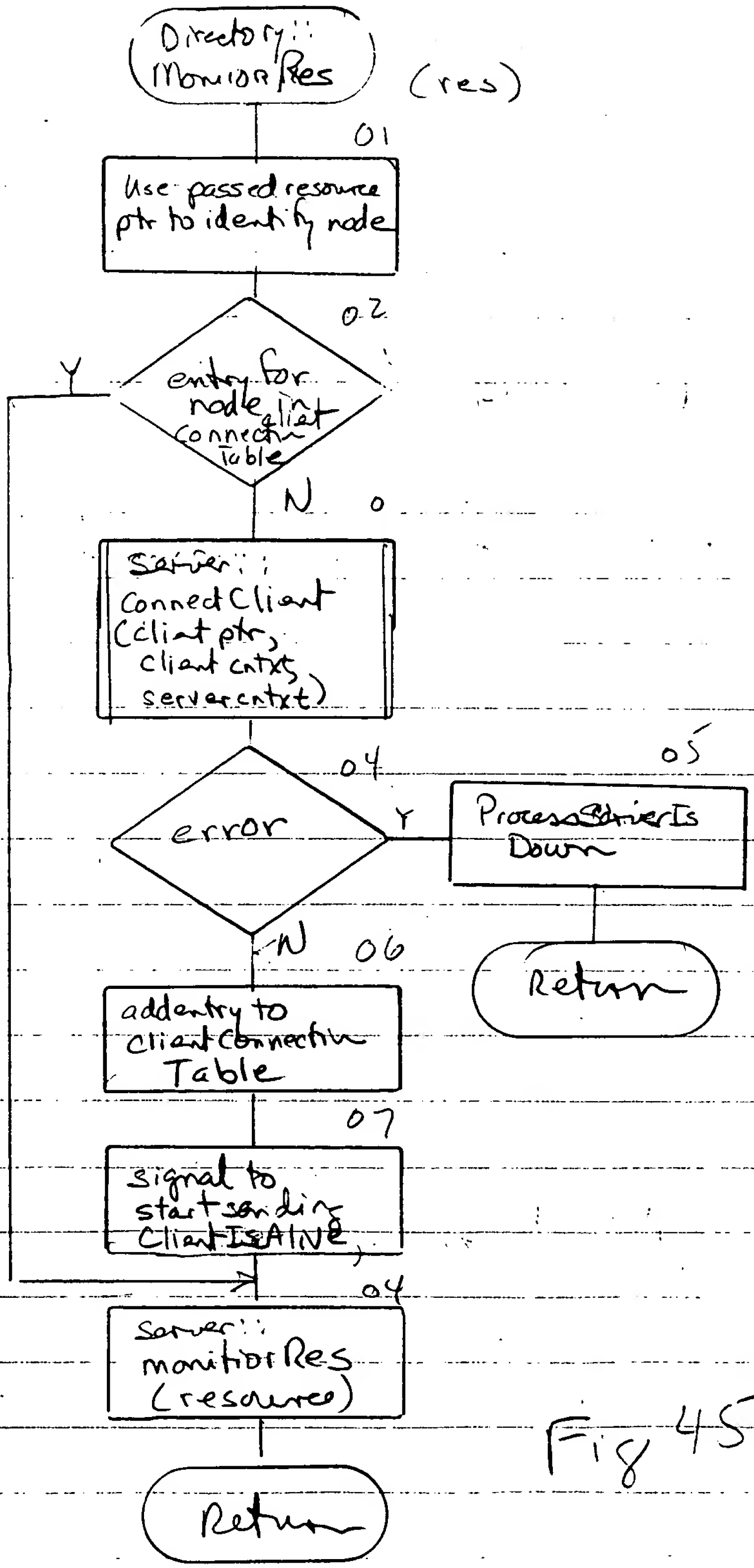

Directory::
MonitorRes
(res)
01
Use passed resource ptr to identify node
02
entry for node in client connection Table
Y
N
0
Server::
ConnectClient
(client ptr,
client cntxt,
server cntxt)
04
error
Y
05
ProcessServerIsDown
Return
N
06
addentry to client Connection Table
07
signal to start sending ClientIsAlive
04
Server::
monitorRes
(resource)
Return

Fig 45

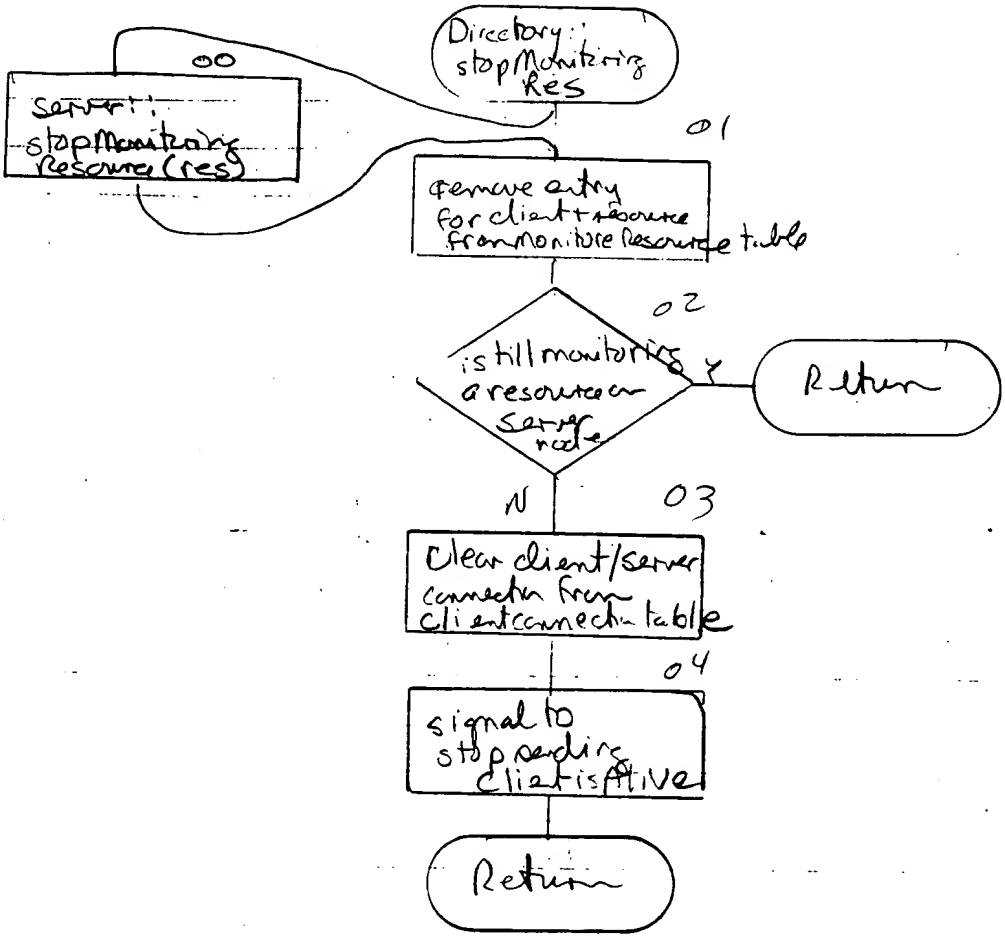

Server: stopMonitoring Resource (res)
00
Directory: stopMonitoring Res
01
remove entry for client + resource from MonitorResource table
02
is till monitoring a resource on Server node
Y
Return
N
03
Clear client/server connection from clientconnection table
04
signal to stop sending ClientisActive
Return

Fig 46

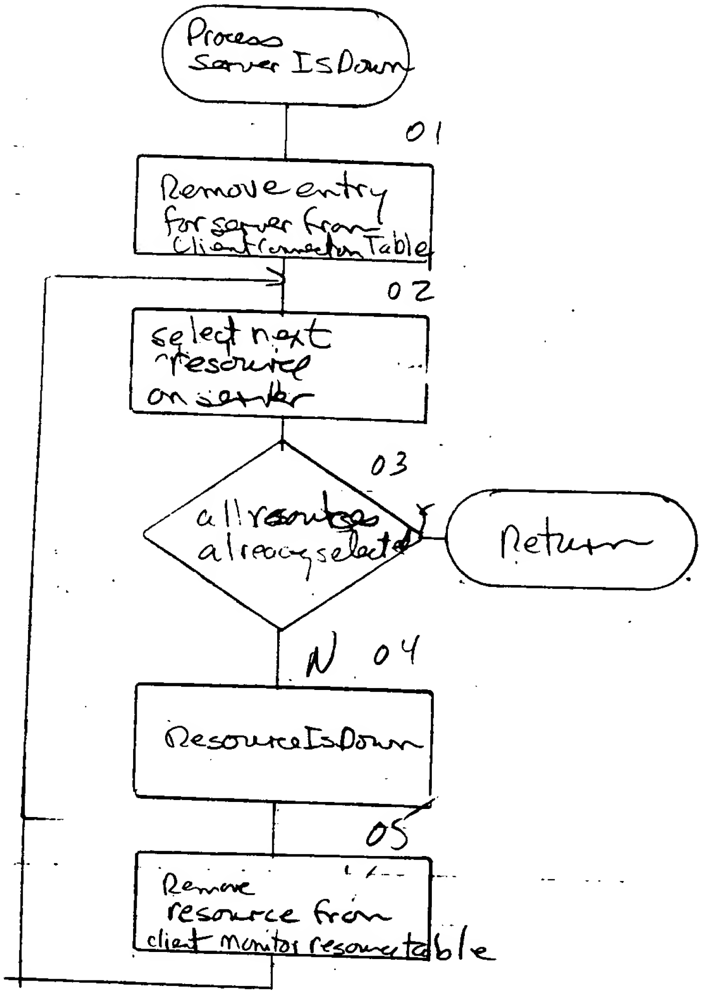

Process ServerIsDown
01
Remove entry for server from ClientConnectionTable
02
select next resource on server
03
all resources already selected
Y
Return
N
04
ResourceIsDown
05
Remove resource from client monitor resource table

Fig 46A

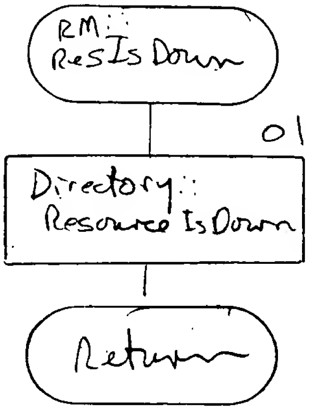
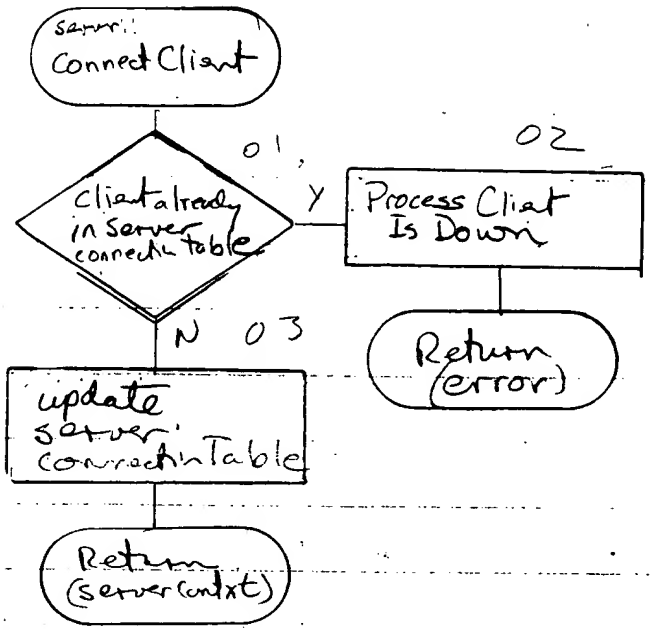


Fig 47


server::
Connect Client
Client already in server connectn table
01
Y
02
Process Cliat Is Down
Return (error)
N
03
update server connectn Table
Return (server contxt)

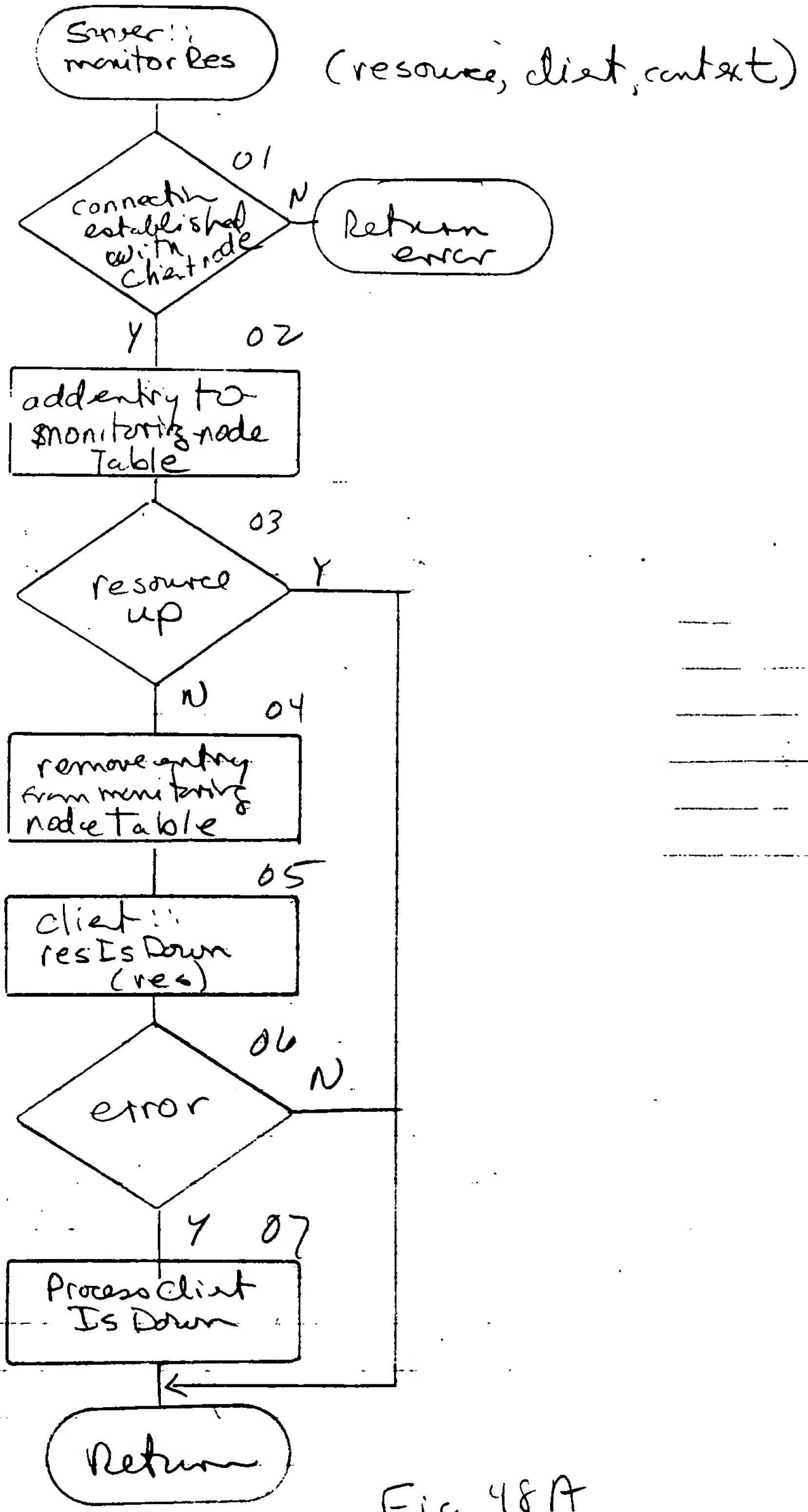

Server::
monitorRes
(resource, client, context)
01
connection established with Client node
N
Return error
Y
02
add entry to $monitoring node Table
03
resource up
Y
N
04
remove entry from monitoring node Table
05
client::
resIsDown
(res)
06
error
N
Y
07
Process Client Is Down
Return

Fig 48A

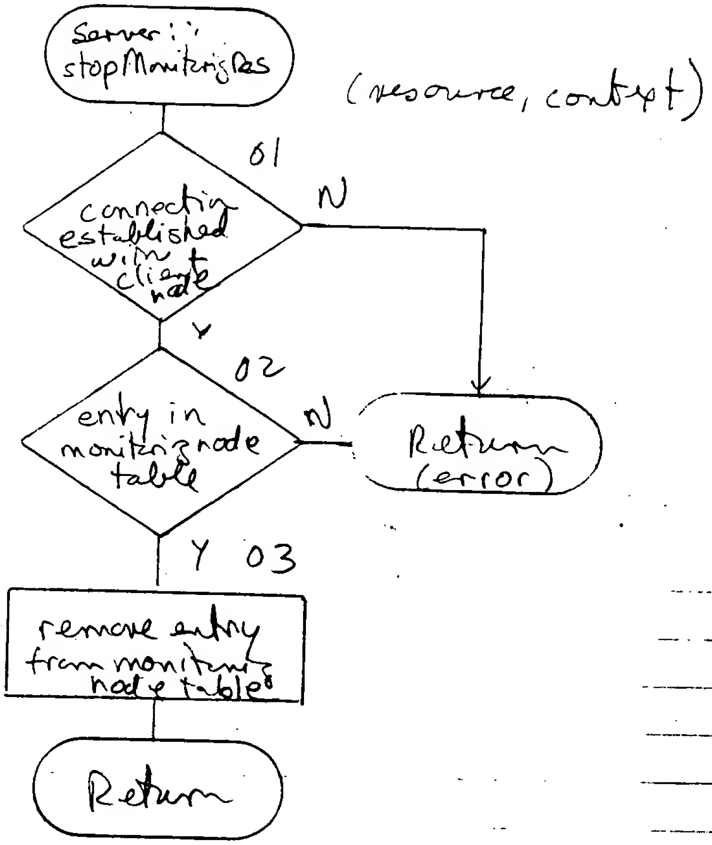
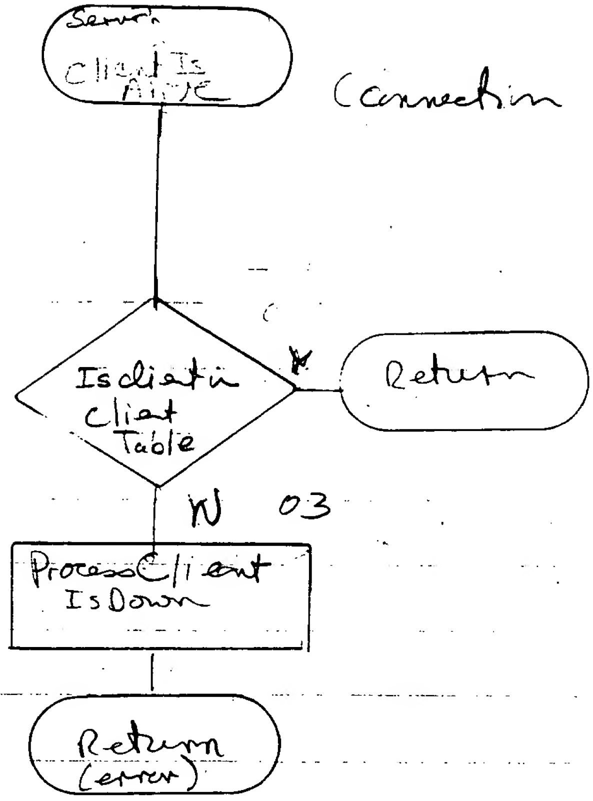
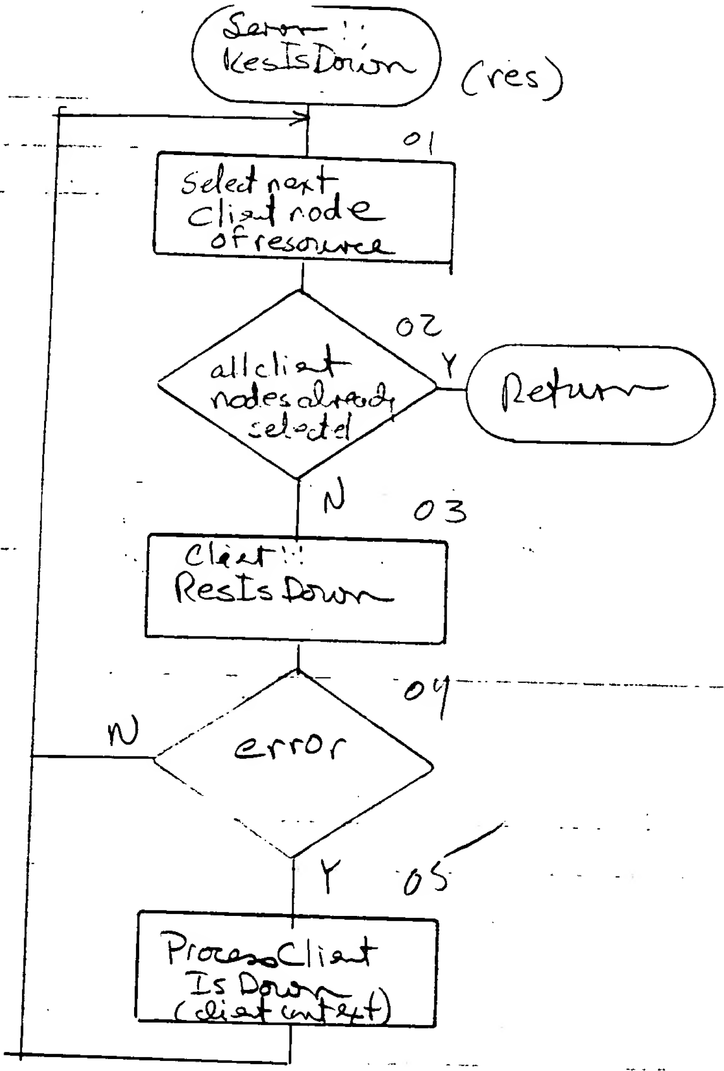
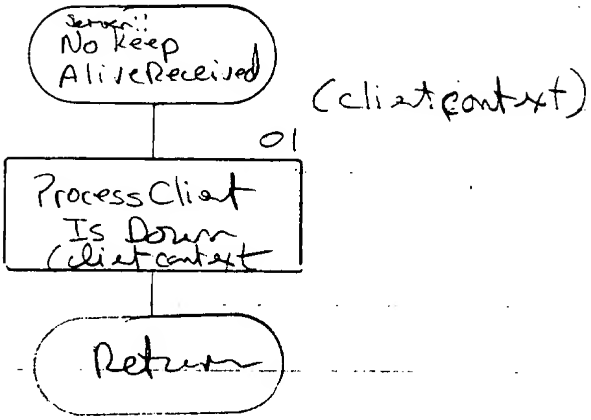
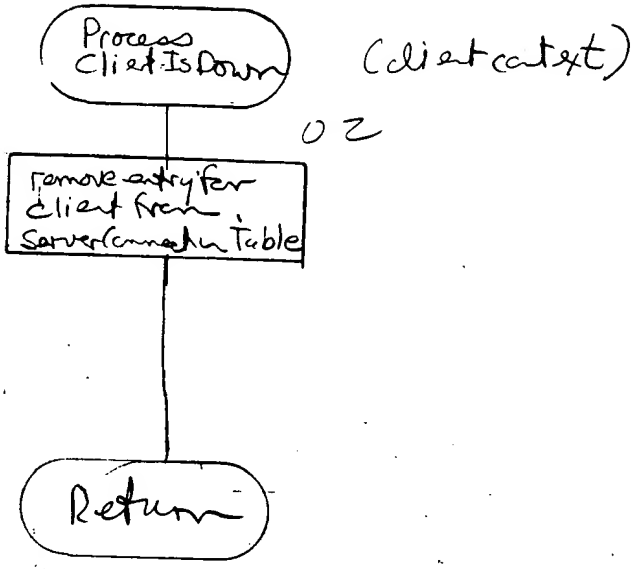

Server::
stopMonitoringRes
(resource, context)
01
connection established with client node
N
Y
02
entry in monitoring node table
N
Return (error)
Y
03
remove entry from monitoring node table
Return

Figure 48B

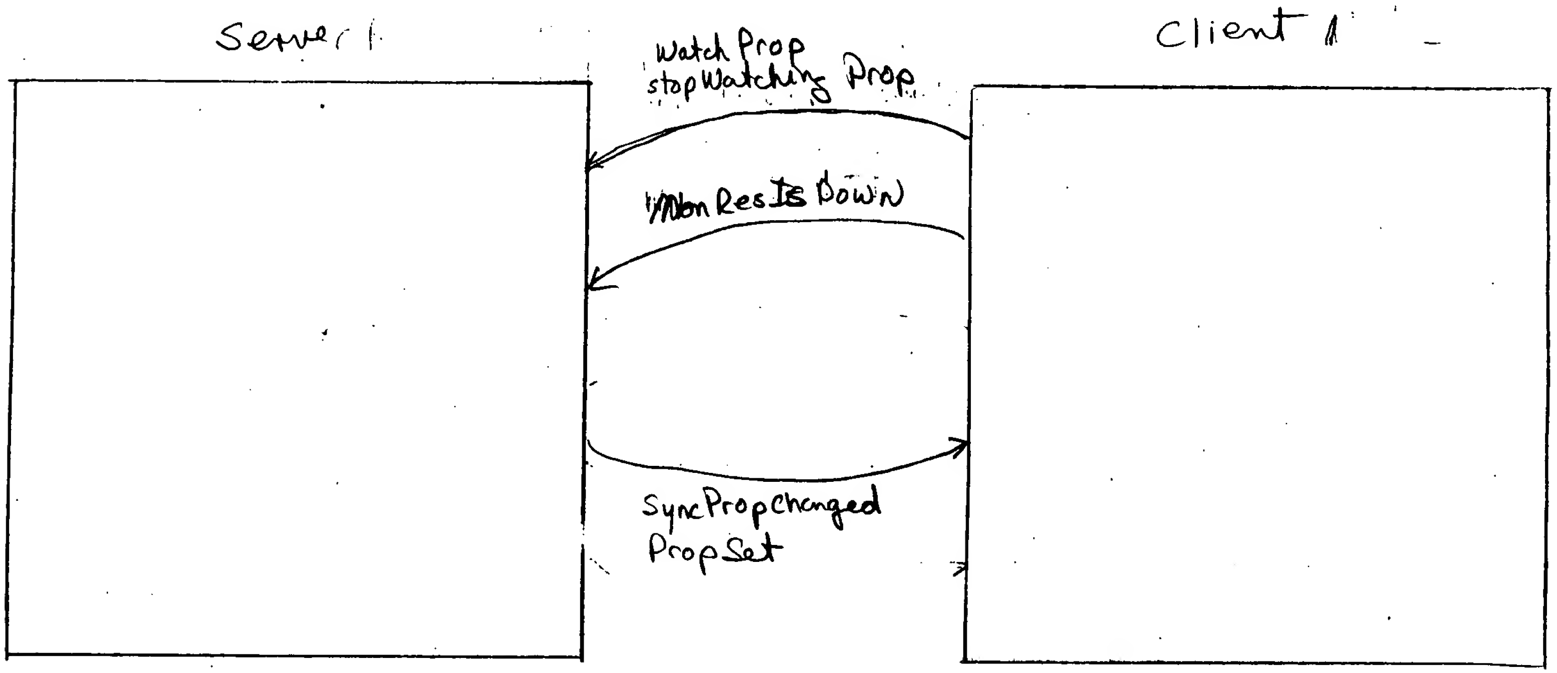

Server's
Client Is Alive
Connection
Is client in Client Table
Y
Return
N
03
Process Client IsDown
Return (error)

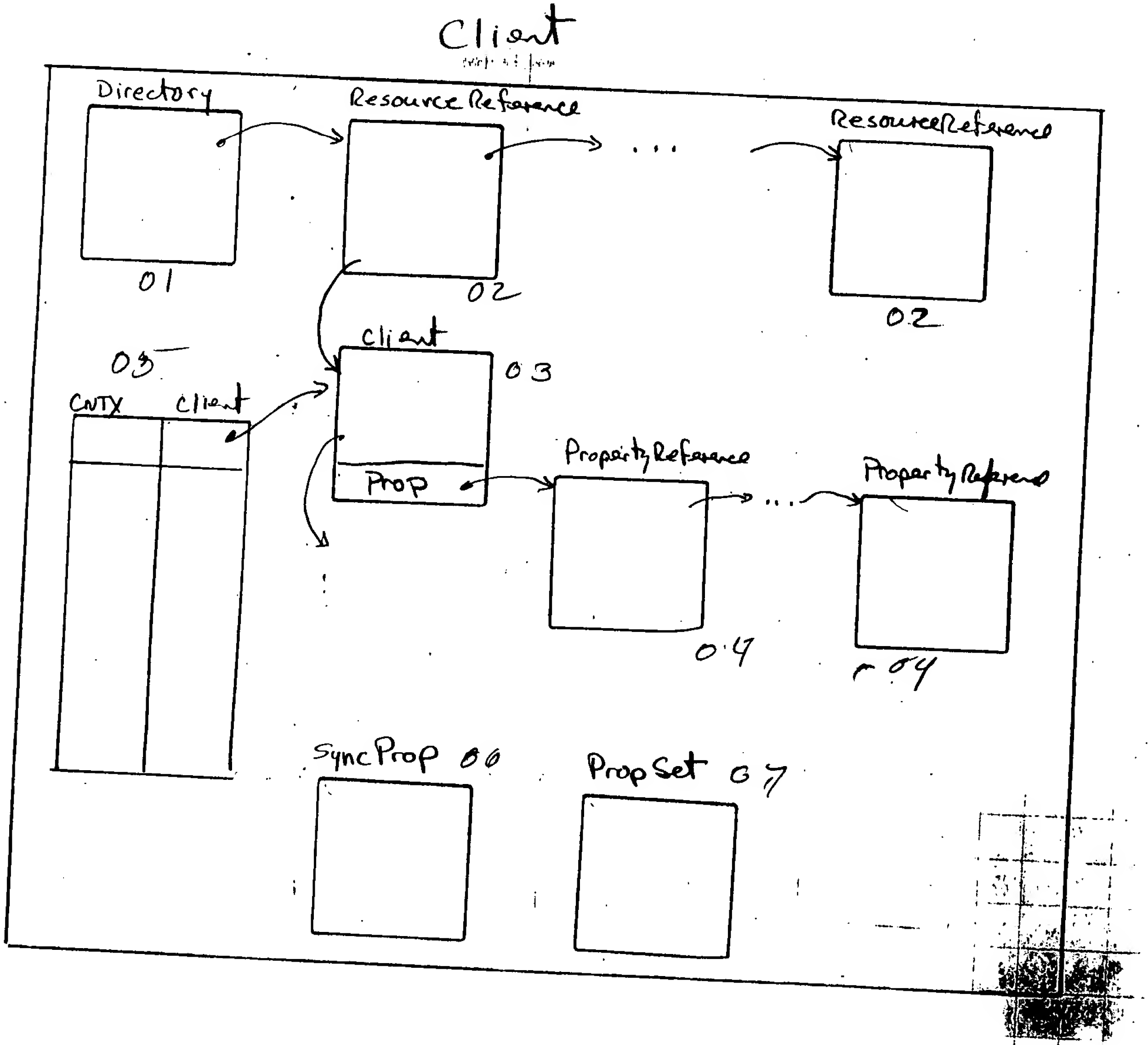

Server::
ResIsDown
(res)
01
Select next
Client node
of resource
02
all client
nodes already
selected
Y
Return
N
03
Client::
ResIsDown
04
N
error
Y
05
ProcessClient
IsDown
(client context)

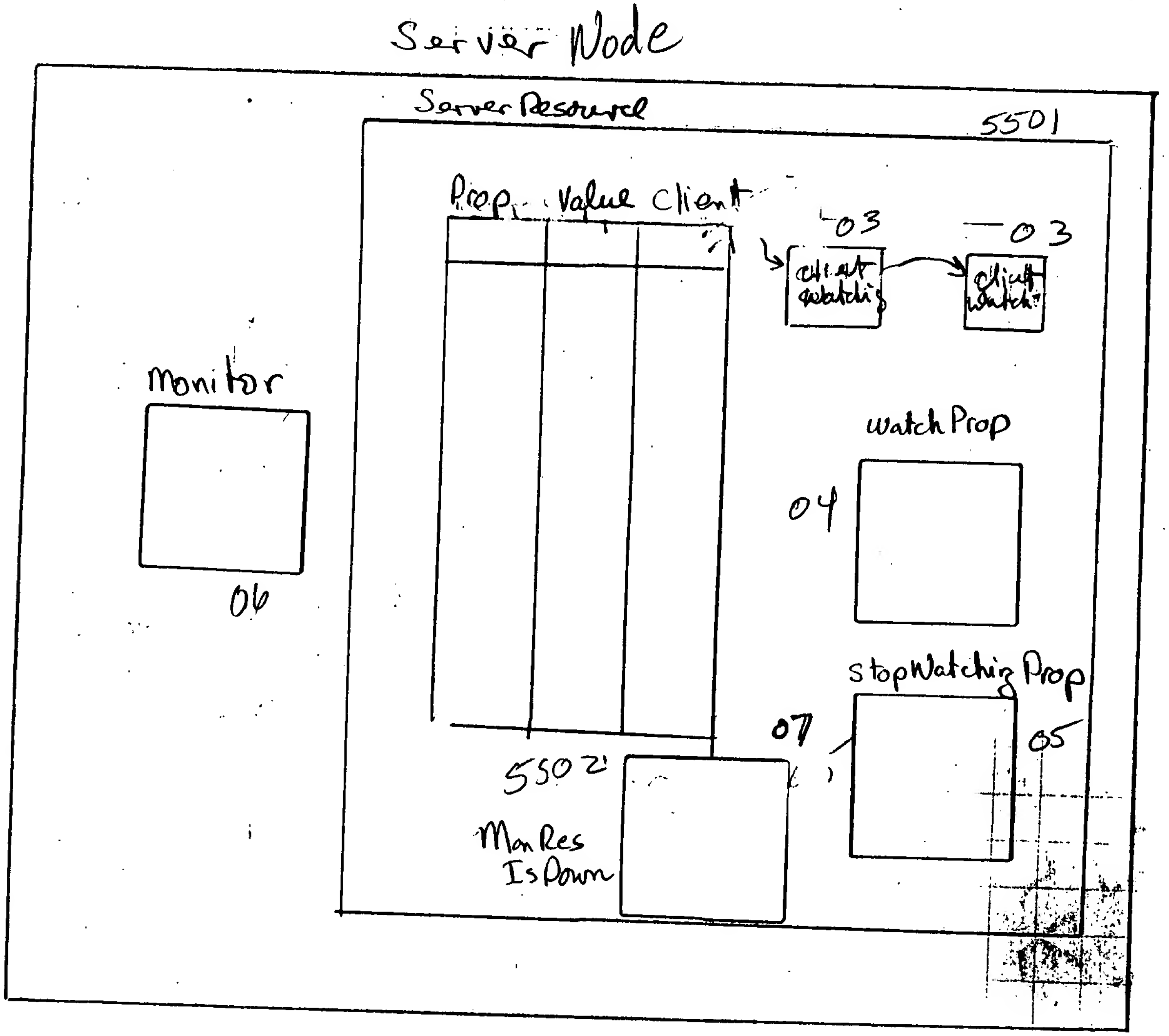

Server::
No Keep
AliveReceived
(clientcontext)
01
ProcessClient
Is Down
(clientcontext
Return

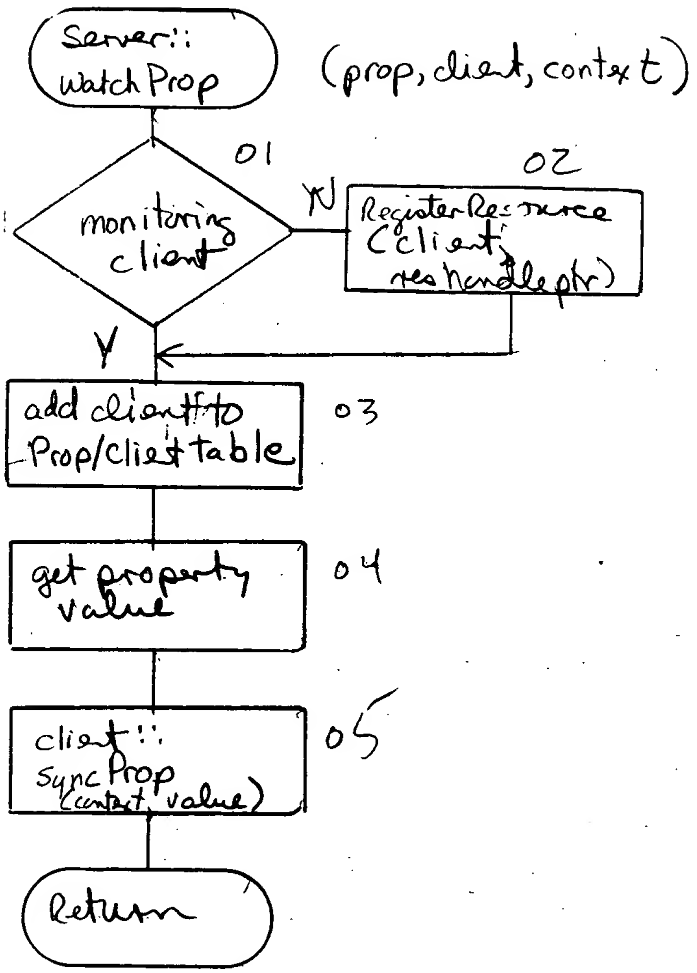

Process ClientIsDown
(client context)
02
remove entry for client from serverConnection Table
Return

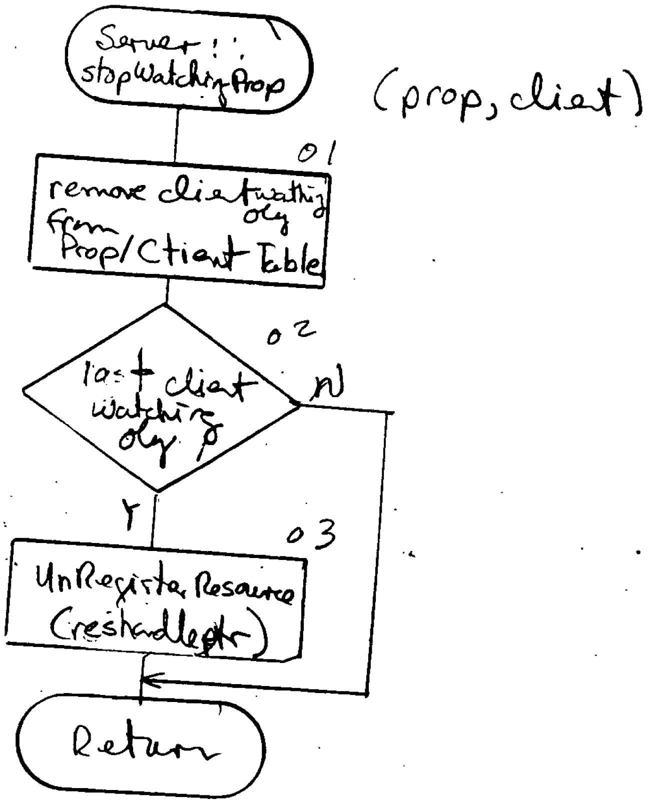
52

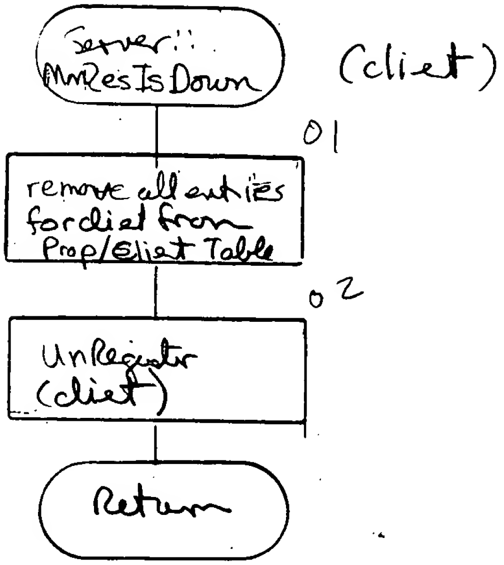

Server 1
Client 1
Watch Prop
stopWatching Prop
MonResIsDown
SyncPropChanged
PropSet

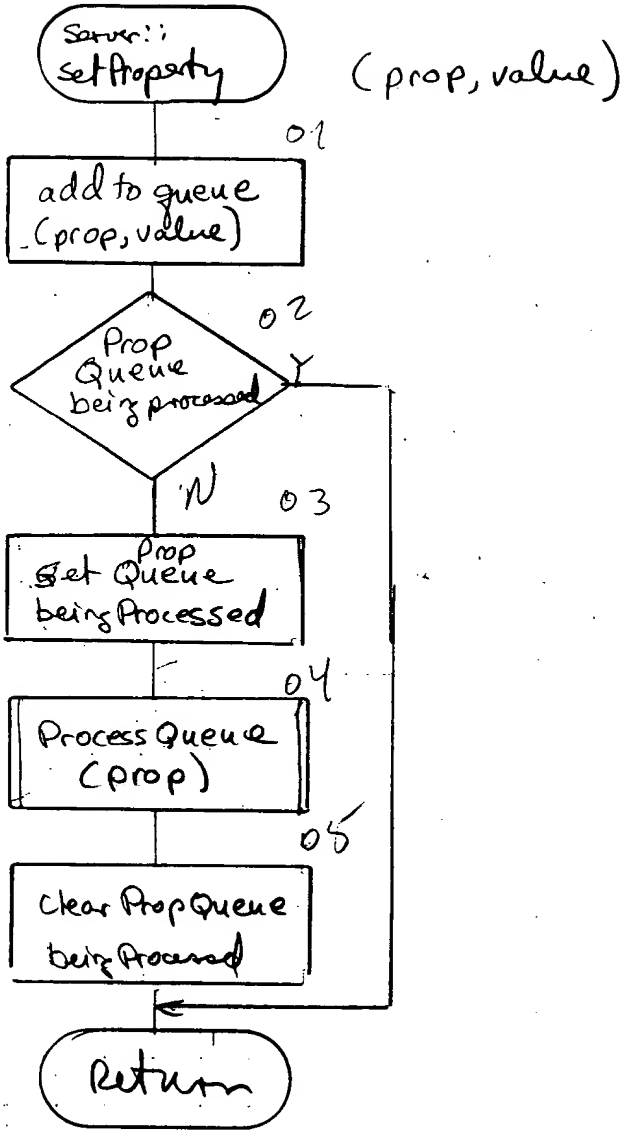

Client
Directory
01
Resource Reference
02
...
ResourceReference
02
03
CNTX
client
client
03
Prop
PropertyReference
04
...
PropertyReference
04
SyncProp 06
PropSet 07

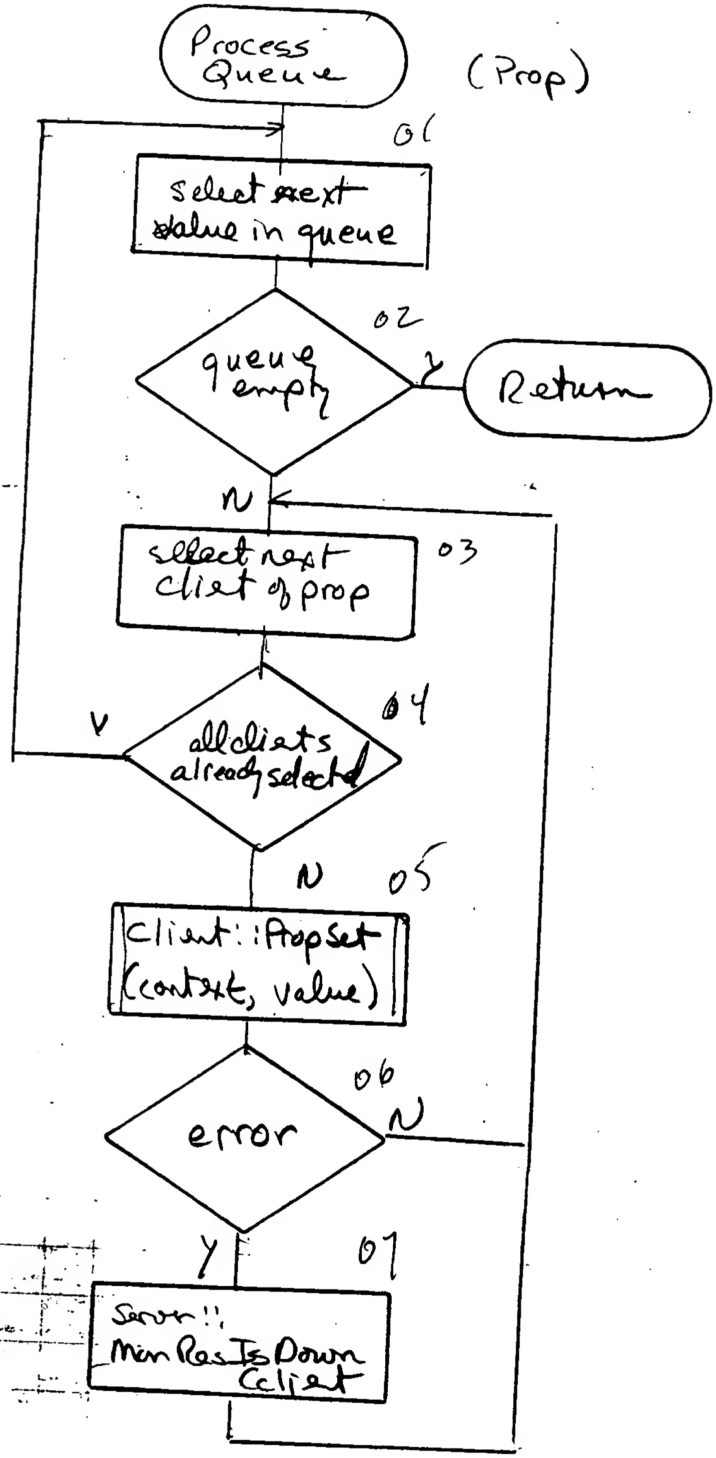

Server Node
Server Resource
5501
Prop. Value Client
03
client watching
03
Client watch
Monitor
06
watchProp
04
stopWatching Prop
07
05
5502
ManRes IsDown

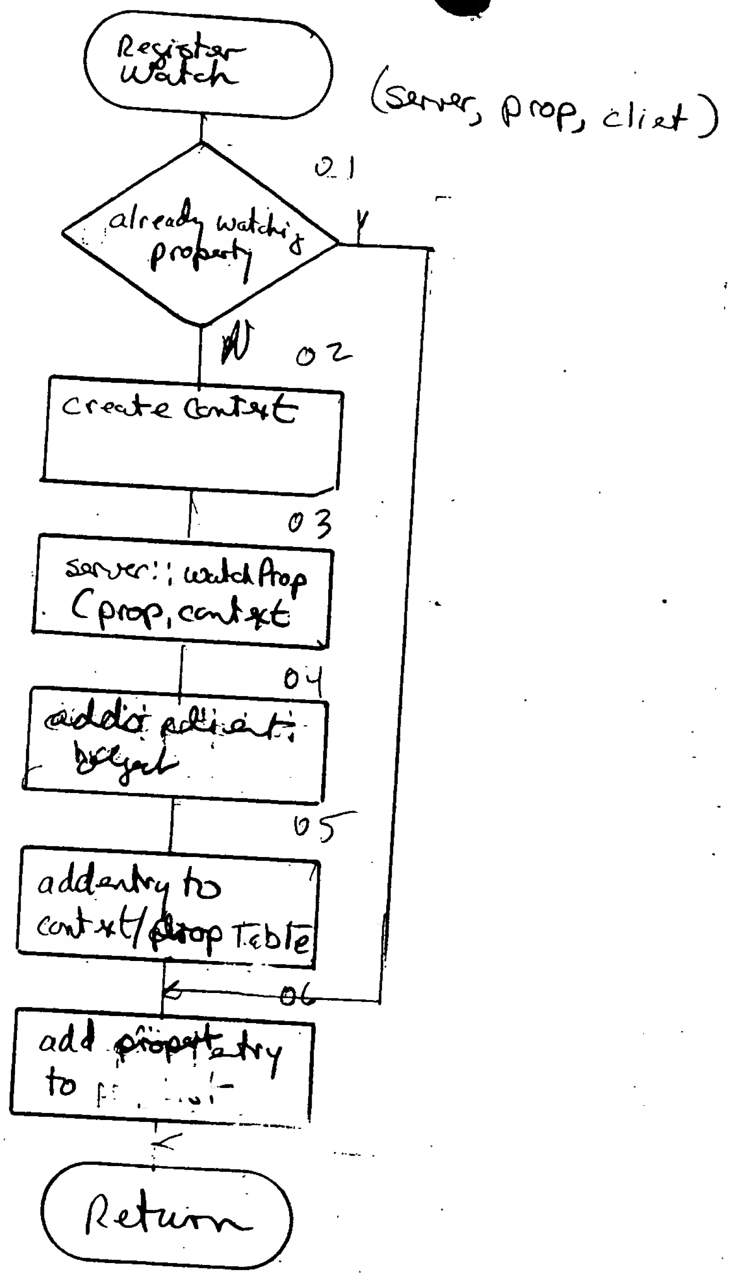

Server::
WatchProp
(prop, client, context)
01
monitoring
client
N
02
RegisterResource
(client,
res handle ptr)
Y
add client to
Prop/Client table
03
get property
value
04
client::
SyncProp
(context, value)
05
Return

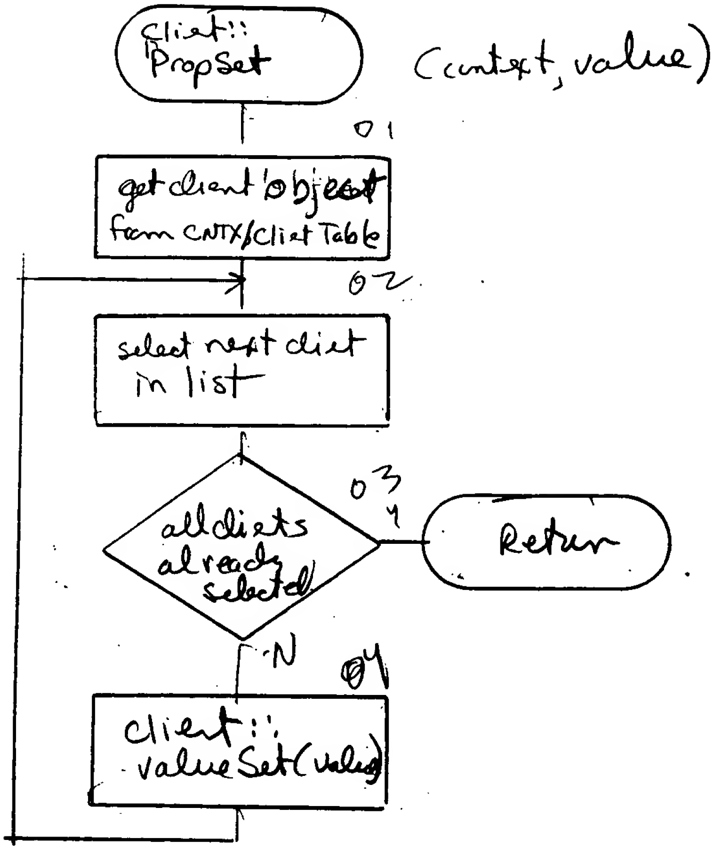

Server::
stopWatchingProp
(prop, cliat)
01
remove cliat watching
obj
from
Prop/Client Table
02
last cliat
watching
obj ?
N
Y
03
UnRegister Resource
(reshandleptr)
Return


Server::
MmResIsDown
(cliet)
01
remove all entries
for cliet from
Prop/Cliet Table
02
UnRegistr
(cliet)
Return


Server::
setProperty
(prop, value)
01
add to queue
(prop, value)
02
Prop
Queue
beig processed
Y
N
03
~~Get~~ Set Prop Queue
being Processed
04
Process Queue
(prop)
05
Clear Prop Queue
beig Processed
Return


Process Queue
(Prop)
01
select ~~next~~ ~~value~~ in queue
02
queue empty
Y
Return
N
03
select next client of prop
04
all clients already selected
Y
N
05
Client::PropSet (context, value)
06
error
N
Y
07
Server::MinResIsDown (client)


Register Watch
(server, prop, client)
01
already watching property
Y
N
02
create Context
03
server::watchProp (prop, context
04
[illegible]
05
addentry to context/~~prop~~ Table
06
add ~~propert~~ entry to [illegible]
Return


Client::
PropSet
(context, value)
01
get client object
from CNTX/Client Table
02
select next client
in list
03
all clients
already
selected
Y
Return
N
04
client::
valueSet(value)

09822157.052800

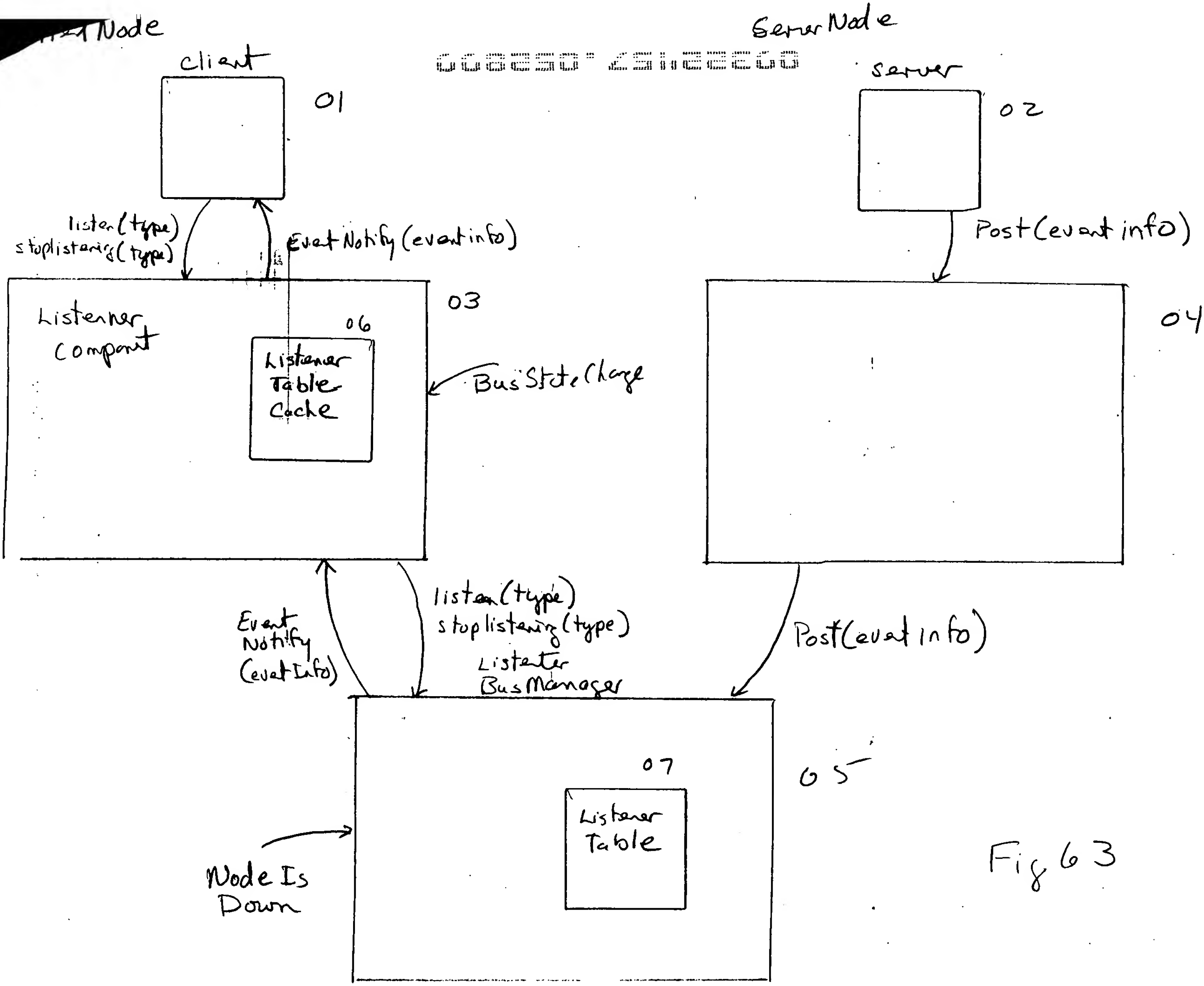


Fig 63